

# < 딥러닝 목차 >

2021년 3월 4일 목요일    오후 4:05

## 12기 딥러닝 수업 목차

단원	번호	상세 목차	이론설명	수업문제
1장 딥러닝 이란?	1	딥러닝으로 할 수 있는 것은?	<a href="#">바로가기</a>	
	2	넘파이(Numpy) 배열 생성하기		
	3	넘파이(Numpy) 산술 연산		
	4	넘파이(Numpy) N차원 배열		
	5	넘파이(Numpy) 브로드캐스트		
	6	넘파이(Numpy) 원소접근		
	7	Matplotlib 로 그래프 그리기		
단원	번호	상세목차	이론설명	수업문제
2장 퍼셉트론	1	퍼셉트론이란 ?	<a href="#">바로가기</a>	
	2	단순한 논리회로		
	3	퍼셉트론 구현하기		
	4	퍼셉트론의 한계		
	5	다층 퍼셉트론 구현하기		
	번호	상세목차	이론설명	수업문제

단원				
3장 신경망	1	퍼셉트론에서 신경망으로		
	2	활성화 함수	<a href="#">바로가기</a>	
	3	다차원 배열의 계산		
	4	3층 신경망 구현하기	<a href="#">바로가기</a>	<a href="#">바로가기</a>
	5	출력층 설계하기		
	6	손글씨 숫자 인식 구현하기	<a href="#">바로가기</a>	
	7	미니배치로 구현하기	<a href="#">바로가기</a>	
단원	번호	상세목차	이론설명	수업문제
4장 신경망 학습	1	손실함수		
	2	수치미분	<a href="#">바로가기</a>	
	3	경사하강법	<a href="#">바로가기</a>	
	4	SimpleNet 구현하기 (단층)	<a href="#">바로가기</a>	<a href="#">바로가기</a> <a href="#">바로가기</a>
	5	수치미분으로 2층 신경망 구현하기	<a href="#">바로가기</a>	<a href="#">바로가기</a> <a href="#">바로가기</a>
단원	번호	상세목차	이론설명	수업문제
5장 오차역전파법	1	계산 그래프	<a href="#">바로가기</a> <a href="#">바로가기</a>	
	2	연쇄법칙	<a href="#">바로가기</a>	
	3	역전파		
	4	단순한 계층 구현하기		<a href="#">바로가기</a>
	5	활성화 함수 계층 구현하기(렐루)	<a href="#">바로가기</a>	
		활성화 함수 계층 구현하기(시그모이드)	<a href="#">바로가기</a>	

	6			
	6	Affine 계층 구현하기	<a href="#">바로가기</a> <a href="#">바로가기</a>	
	7	Softmax 계층 구현하기	<a href="#">바로가기</a>	<a href="#">바로가기</a>
	8	Orderdictionary	<a href="#">바로가기</a>	
	9	오차 역전파법 구현하기	<a href="#">바로가기</a>	
	기타	자코비안 행렬	<a href="#">바로가기</a>	
단원	번호	상세목차	이론설명	수업문제
6장 학습 관련 기술들	1	매개변수 갱신	<a href="#">바로가기</a>	<a href="#">바로가기</a>
	2	가중치 초기값	<a href="#">바로가기</a>	<a href="#">바로가기</a> <a href="#">바로가기</a>
	3	배치정규화	<a href="#">바로가기</a>	
	4	드롭아웃	<a href="#">바로가기</a>	<a href="#">바로가기</a> <a href="#">바로가기</a>
	5	keras 가상환경 만들기	<a href="#">바로가기</a>	
	6	keras 의 early stopping 기능 구현하기	<a href="#">바로가기</a>	<a href="#">바로가기</a>
단원	번호	상세목차	이론설명	수업문제
7장 합성곱 신경망	1	완전 연결 계층의 문제점	<a href="#">바로가기</a>	
	2	합성곱 연산	<a href="#">바로가기</a>	<a href="#">바로가기</a>
	3	패딩	<a href="#">바로가기</a>	
	4	스트라이드	<a href="#">바로가기</a>	
	5	3차원 합성곱 연산 첫번째	<a href="#">바로가기</a> <sup>1</sup> <a href="#">바로가기</a> <sup>2</sup>	<a href="#">바로가기</a> <sup>1</sup> <a href="#">바로가기</a> <sup>2</sup>
	6	3차원 합성곱 연산 두번째	<a href="#">바로가기</a>	<a href="#">바로가기</a>
	7	블록으로 생각하기	<a href="#">바로가기</a>	
	8	배치처리	<a href="#">바로가기</a>	
	9	풀링계층	<a href="#">바로가기</a>	
	10	im2col 함수	<a href="#">바로가기</a> <a href="#">바로가기</a>	<a href="#">바로가기</a>

	11	CNN 구현하기	<a href="#">바로가기</a>	
	12	CNN 전체코드	<a href="#">바로가기</a>	
	13	최종 스크립트의 큰 그림	<a href="#">바로가기</a>	
	14	고전 CNN LeNet 구현하기	<a href="#">바로가기</a>	
<b>단원</b>	<b>번호</b>	<b>상세목차</b>	<b>이론설명</b>	<b>수업문제</b>
8장 텐서플로우	1	텐서 플로우란?	<a href="#">바로가기</a>	
	2	텐서 플로우 환경구성 (Tensorflow 1.x)	<a href="#">바로가기</a>	
	3	텐서 플로우 기본 문법		
	4	넘파이와 텐서 플로우 비교		
	5	Mnist 로 단층 신경망 구현하기		
	6	텐서 플로우로 구현하는 비용함수		
	7	경사감소법을 텐서 플로우로 구현	<a href="#">바로가기</a>	
	8	텐서 플로우로 다층 신경망 구현하기		
	9	텐서 플로우로 CNN 신경망 구현하기		
	10	딥러닝의 역사	<a href="#">바로가기</a>	
	11	신경망에 데이터 로드 함수들 (cifar)	<a href="#">바로가기</a>	
	12	신경망에 데이터 로드 함수들 (이파리)	<a href="#">바로가기</a>	
	13	사진을 32x32로 resize 하는 함수	<a href="#">바로가기</a>	
	14	GPU 에 CUDA / cuDNN 설치 하는 방법	<a href="#">바로가기</a>	<a href="#">확인방법</a>
	15	Tensorflow 1.x 버전 + keras 2.x 설치	<a href="#">바로가기</a>	<a href="#">GPU 확인</a>
	16	Tensorflow 1.x 버전 + keras 2.x 에서 이파리 데이터 분류 신경망 구현하기	<a href="#">바로가기</a>	<a href="#">1.x 와 2.x 비교하기</a>
	17	Tensorflow 1.x 버전 + keras 2.x 에서 이파리 데이터 모델 불러오는 코드	<a href="#">바로가기</a>	
	18	Tensorflow 2.x 환경구성	<a href="#">바로가기</a>	
	19	Tensorflow 2.x 에서 퍼셉트론 구현	<a href="#">바로가기</a>	
	20	Tensorflow 2.x 에서 mnist 신경망	<a href="#">바로가기</a>	
	21	keras 첫번째 층에 input_shape	<a href="#">바로가기</a>	
	22	Tensorflow 2.x 에서 CNN 안쓴 신경망	<a href="#">바로가기</a>	<a href="#">드롭아웃</a>

	23	Tensorflow 2.x 에서 CNN 사용 신경망	<a href="#">바로가기</a>	<a href="#">이파리</a>
	24	Tensorflow 2.x 에서의 이미지 증식	<a href="#">바로가기</a>	
	25	Tensorflow 2.x 에서의 전이학습	<a href="#">바로가기</a>	
	26	이미지넷 우승 CNN - VGG 구현하기	<a href="#">바로가기</a>	<a href="#">바로가기</a>
	27	이미지넷 우승 CNN - ResNet 구현하기	<a href="#">바로가기</a>	
	28	이미지 웹스크롤링 코드 총정리	<a href="#">바로가기</a>	
	29	신경망 활용 홈페이지 만들기	<a href="#">바로가기</a>	<a href="#">바로가기</a> <a href="#">이파리</a>
	30	Obejct Detection	<a href="#">바로가기</a>	<a href="#">바로가기</a> <a href="#">바로가기</a>
	31	Gan 으로 이미지 생성	<a href="#">바로가기</a>	<a href="#">바로가기</a> 1 <a href="#">바로가기</a> 2
	32	미술작품 스타일로 일반 사진 변환	<a href="#">바로가기</a>	<a href="#">환경구성</a> <a href="#">구현</a>

아나콘다 완전 삭제 방법: <http://cafe.daum.net/oracleoracle/Sedp/524>

**딥러닝을 배웠으면 아는 걸로 끝나지 말고 ... 뭔가 필요한걸 만들어보자** [바로가기](#)

딥러닝책 :

O'REILLY

파이썬으로 익히는 딥러닝 이론과 구현

# Deep Learning

from Scratch

밑바닥부터 시작하는 딥러닝



한빛미디어

사이토 고키 지음  
계명영시 옮김

# 03/04

2021년 3월 4일 목요일 오전 9:49

## ■ 머신러닝과 딥러닝의 차이 간단하게 설명

<https://cafe.daum.net/oracleoracle/SgRM/3>

머신러닝으로 기계학습을 하려면 정형화된 데이터가 있어야 했다. 표 형태의 정형화된 데이터를 만들려고 판다스를 이용했다. 이런 정형화된 데이터를 만드는 것이 상당한 노동력이 들어가는 일이다. 그래서 딥러닝은 그냥 이미지를 신경망에 보여주기만 하면된다. 그러면 스스로 학습을 해서 그 이미지를 신경망이 알아볼 수 있는 상태까지 되는 것이다.

## ■ 딥러닝 기술을 이용해서 현업에서 하고 있는 일들?

1. 인천공항에 컨테이너 검색대에 물건을 올리면 CNN으로 판별하여 위반되는 물품이 있는지 검사하는 신경망을 개발
2. 의료쪽에서는 X-ray 사진과 의료영상 사진의 질병여부를 컴퓨터가 판별
3. 물품(의류)의 불량품을 판별하는 신경망 개발 + 인터페이스
4. 주가 예측 신경망 개발
5. 외국어 번역하는 신경망
6. 인공지능 변호사 ( 법률책 ---> 신경망 ---> 판결 )
7. 음악 작곡 신경망 (Gan) ---> 아마존 키보드

## ■ 넘파이( numpy ) 배열 생성하기

numpy란 ?

python 언어에서 기본적으로 지원하지 않는 배열( array ) 혹은 행렬( matrix )의 계산을 쉽게 해주는 라이브러리

머신러닝에서 많이 사용하는 선형대수학에 관련된 수식들을 python에서 쉽게 프로그래밍 할 수 있게 해준다.

### 문제1. 아래의 행렬을 numpy로 만드시오!

```
[ 1  2  
 3  4]
```

답 :

```
import numpy as np
```

```
a = np.array([[1,2],[3,4]])  
print(a)
```

### 문제2. 위의 a행렬의 각 요소에 숫자 5를 더하시오!

답 :

```
import numpy as np
```

```
a = np.array([[1,2],[3,4]])  
print(a + 5)
```

결과 :

```
[[6 7]  
 [8 9]]
```

### 문제3. 아래의 두 행렬의 합을 구하시오!

답 :

```
import numpy as np
```

```
a = np.array([[1,5],[6,7]])  
b = np.array([[3,4], [2,1]])
```

```
print(a+b)
```

결과 :

```
[[4 9]  
 [8 8]]
```

※ **numpy의 유용한 기능 중 하나인 브로드캐스트 ( broadcast ) ?**

넘파이에서는 형상이 다른 배열끼리도 계산을 할 수 있다.



아래의 경우처럼 숫자 5가 2x2행렬로 확대된 후 연산이 이루어 지는 것을 브로드 캐스트라고 한다.

**문제4. 아래의 브로드캐스트 기능을 numpy로 구현하시오!**



답 :

```
import numpy as np

a = np.array([[1,2],[3,4]])
print(a *10)
```

**문제5. 아래의 그림 1-2도 브로드캐스트 되는지 테스트 하시오!**



답 :

```
import numpy as np

a = np.array([[1,2],[3,4]])
b = np.array([[10,20]])

print(a*b)
```

**문제6. 아래의 신경망을 numpy 행렬로 구현하시오!**



답 :

```
x = np.array([[2,4,8]])  
w = np.array([[4,3,2]])  
k= np.sum(x*w)
```

```
print(k)
```

문제7. 아래의 신경망을 numpy 행렬로 구현하시오! ( 힌트 : `np.dot(x,w)` )



답 :

```
x = np.array([[1,2]])  
w = np.array([[1,3,5],[2,4,6]])  
k = np.dot(x,w)  
print(k)
```

문제8. 아래의 신경망을 numpy로 구현하시오!



답 :

```
x = np.array([1,2]) # 입력값 행렬
w1 = np.array([[1,3,5],[2,4,6]]) # 가중치 행렬1
w2 = np.array([[3,4],[5,6],[7,8]]) #가중치 행렬2
k = np.dot(x,w1) # 1층
m = np.dot(k,w2) # 2층
print(m) # [189 222]
```

## ■ 4. 넘파이(Numpy) N차원 배열 ( p38 )

넘파이는 1차원 배열( 1줄로 늘어선 배열 ) 뿐만 아니라 다차원 배열로도 작성할 수 있다.

예제 :

```
import numpy as np
x = np.array( [[1,2], [3,4] ] )
print(x.shape)
```

**문제9. 아래와 같이 5x5행렬을 생성하시오!**



답 :

```
x = np.array([[1,2,3,4,5], [2,4,3,2,4], [3,1,4,2,1], [2,7,3,5,4], [1,5,6,3,1]])
print(x.shpae ) #( 5,5 )
```

※ 넘파이 배열( np.array )은 N차원 배열을 작성할 수 있다.

1차원 배열, 2차원 배열, 3차원 배열처럼 원하는 차수의 배열을 만들 수 있다.

수학에서는 1차원 배열을 벡터( vector )라고 하고 2차원 배열을 행렬( matrix )라고 부른다.

또 벡터와 행렬을 일반화한 것을 텐서( tensor )라고 한다.

tensor flow -----> 행렬이 계산되면서 흘러간다.

↓     ↓

다차원, 흐름

구글에서 만든 텐서플로우는 다차원 배열의 연산( 계산 )을 빠르게 할 수 있게끔 구현이 되어져 있다.

1. 코드가 간결하다.
2. 신경망 구현에 필요한 모든 함수들이 다 내장되어 있다.
3. 속도가 아주 빠르다. ( 코딩, 실행 모두 빠르다 )
4. GPU를 사용할 수 있다.

문제10. ( 점심시간 문제 ) 아래의 행렬의 내적을 구현하시오!



```
x = np.array([[3,4,2], [4,1,3]])
y = np.array([[1,5], [2,3], [4,1]])
k = np.dot(x,y)
print(k)
```

## ■ 6. 넘파이( Numpy ) 원소접근 ( p 40 )

numpy 배열안에 요소들에 대한 접근은 numpy를 이용하지 않았을 때보다 훨씬 간단하게 구현할 수 있다.

문제11. 아래의 리스트를 만들고 아래의 리스트에서 15이상인 숫자들만 출력하시오!

**[ 51, 55, 14, 19, 0, 4 ]**

답 :

```
a = np.array( [ 51, 55, 14, 19, 0, 4 ] )
b = []
for i in a:
    if i >=15 :
        b.append(i)

print(b)
```

**문제12. 아래의 행렬식을 만들고 아래의 행렬의 요소에서 15 이상인것만 출력하십시오!**

**( numpy를 이용하지 않고 수행하세요 )**

**51 55  
14 19  
0 4**

답 :

```
a = [ [51,55], [14,19], [0,4] ]
b = []
for i in a:
    for j in i:
        if j >= 15 :
            b.append(j)

print(b)
```

**문제13. 위의 예제를 numpy를 이용해서 구현하십시오!**

```
import numpy as np
a = [ [51,55], [14,19], [0,4] ]
b = np.array(a)

print(b[b>=15]) # 설명 : b[ 조건 ]
```

설명 : numpy를 이용하면 코딩을 어렵게 안해도 된다. 쉽고 빠르게 원하는 것을 구현할 수 있다.

**문제14. 아래의 행렬을 만들고 아래의 행렬에서 3 이상인 것만 출력하십시오!**

5.1 3.5 1.4 0.2

```
4.9 3 1.4 0.2
4.7 3.2 1.3 0.2
4.6 3.1 1.5 0.2
```

답 :

```
a = [[5.1, 4.9, 4.7, 4.6], [3.5, 3, 3.2, 3.1], [1.4, 1.4, 1.3, 1.5], [0.2, 0.2, 0.2, 0.2]]
b = np.array(a)
print(b[b>=3])
```

## ■ 7. Matplotlib로 그래프 그리기

딥러닝 실험에서는 그래프 그리기와 데이터 시각화가 중요하다. Matplotlib는 그래프를 그리기 위한 라이브러리 이다.

matplotlib를 이용하면 그래프가 그리기 쉬워진다. 신경망에서 사용하는 그래프는 주로 라인 그래프이다. 정확도가 점점 올라가는지 아니면 에러(오차)가 점점 떨어지는지 확인할 때 유용하다.

### 예제1. 파이썬으로 산포도 그래프를 그리는 방법

```
x = np.array( [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] )
y = np.array( [9, 8, 7, 9, 8, 3, 2, 4, 3, 4] )

import numpy as np
import matplotlib.pyplot as plt

plt.scatter( x, y, color='red', s=80 ) # s는 점사이즈이다.
plt.show()
```



문제15. 위의 그래프를 라인 그래프로 그리시오!

```
x = np.array( [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] )
y = np.array( [9, 8, 7, 9, 8, 3, 2, 4, 3, 4] )

import numpy as np
import matplotlib.pyplot as plt

plt.plot( x, y, color='red' ) # 산포도일땐 scatter, 라인일때는 plot
plt.show()
```



**문제16. 위의 그래프에 제목을 "신경망 오차 그래프"라고 하시오!**

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import font_manager, rc

font_path = "d:\wwdata\wwmalgun.ttf" #폰트파일의 위치 실습파일안에있음
font_name = font_manager.FontProperties(fname=font_path).get_name()
rc('font', family=font_name)

x = np.array( [ 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 ] )
y = np.array( [ 9, 8, 7, 9, 8, 3, 2, 4, 3, 4 ] )

plt.plot( x, y, color='red' )
plt.title("신경망 오차 그래프")
plt.show()
```

## ■ 1장의 내용 정리

**1. 딥러닝이 무엇인가? ( 면접질문 중 가장 많이 물어보는 질문 )**

여러 비선형 변환 기법의 조합을 통해 높은 수준의 추상화를 시도하는 기계학습 알고리즘의 집합이다.

**추상화** : 다량의 데이터나 복잡한 자료들 속에서 핵심적인 내용 또는 기능을 요약하는 기법

**딥러닝이 부각 된 이유 ?**

1. 기존 인경신경망 모델의 문제점인 기울기 소실이 해결되었다.
2. 강력한 GPU 연산을 활용하여 하드웨어 연산속도를 높였다.  
(구글 코랩 이용하면 GPU를 무료로 사용 가능하다.)
3. 빅데이터의 등장과 SNS의 활용이 증가하여 학습에 필요한 데이터 확보가 가능해졌다.

## 2. 신경망을 구현할 때 numpy를 왜 사용하는 것인가?

행렬 연산을 빠르게 할 수 있는 기능이 numpy에 내장되어 있는데 신경망에 데이터를 입력하면 행렬연산이 일어나고 맨 마지막으로 확률이 출력이 된다. 이 사진이 얼룩말인지 제규어인지에 대한 확률이 마지막에 출력되기 위해서는 다차원 행렬 계산을 해야합니다.

## 3. numpy의 주요 기능중에서 broadcast는 무엇인가?

형상이 다른 배열끼리 계산을 스마트하게 할 수 있게 해주는 numpy의 기능이다.

## 4. 딥러닝 실험을 위해 시각화가 필요한 이유가 무엇인가 ?

신경망이 제대로 학습이 되고 있는지 확인하려면 학습과정을 시각화 해보아야 한다.

# ■ 2장. 퍼셉트론

■ 퍼셉트론( perceptron )이란 ?

- 인간의 뇌세포 하나를 컴퓨터로 흉내낸 것
- 1957년에 프랑크 로젠블라트가 퍼셉트론을 고안했다.
- 사람의 뇌의 동작을 전기 스위치 온/오프로 흉내낼 수 있다는 이론을 증명했다.



퍼셉트론을 간단히 얘기하면 인간의 신경세포 하나를 흉내를 냈는데 고등학교 생물 시간에 배운 3가지 용어 구현

1. 자극 ( stimulus )
2. 반응 ( response )
3. 역치 ( threshold )

" 특정 자극이 있다면 그 자극이 어느 역치 이상이어야 세포가 반응한다 "

예 : 짜게 먹는 사람은 자기가 평소에 먹는 만큼 음식이 짜지 않으면 싱겁다고 느낀다.

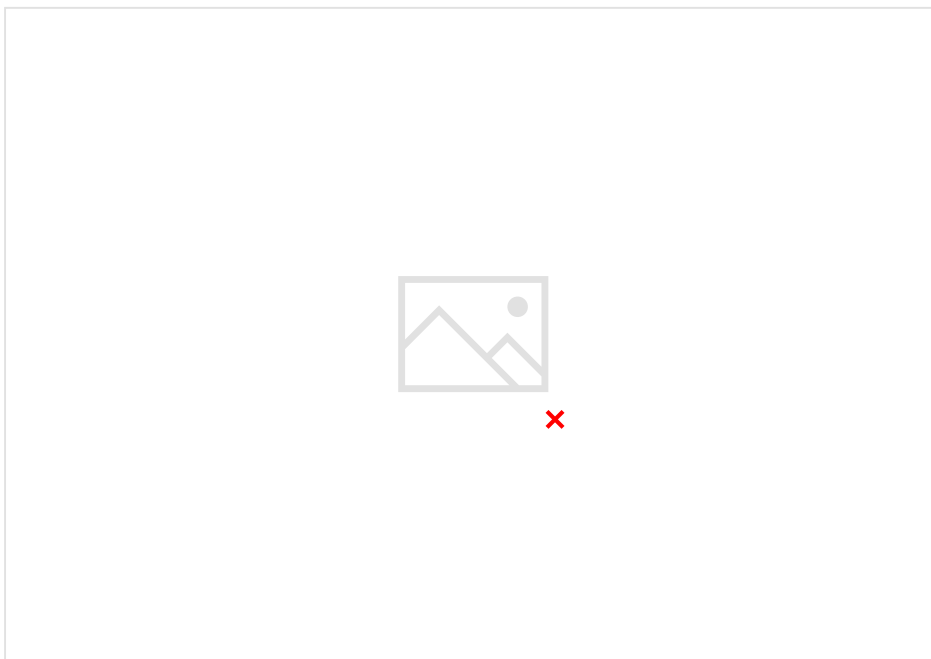
( 역치 이하의 자극은 무시 )

### ■ 단순한 논리회로 ( p 49 )

1. AND 게이트
2. OR 게이트
3. NAND 게이트
4. XOR 게이트

### 罏 AND 게이트

그림 2-2



문제17. 위의 AND 게이트를 위한 데이터 행렬을 numpy array로 구현하시오!

```
X = np.array( [[0,0], [0,1], [1,0], [1,1]] ) #입력 데이터
```

```
y = np.array( [ [0], [0], [0], [1] ] ) # 정답
```

```
print( X.shape )
```

```
print( y.shape )
```

### 문제18. AND 게이트 퍼셉트론 함수를 생성하시오!

```
def AND( x1, x2 ):
```

```
    w1, w2, theta = 0.5, 0.5, 0.7
```

```
    tmp = x1*w1 + x2 * w2 #입력값과 가중치의 곱의 총합
```

```
    if tmp <= theta: # 입력값과 가중치의 총합이 임계치를 넘지 않는다면
```

```
        return 0
```

```
    elif tmp > theta:
```

```
        return 1
```

```
print(AND(0,0)) #0
```

```
print(AND(1,0)) #0
```

```
print(AND(0,1)) #0
```

```
print(AND(1,1)) #1
```

### 문제19. OR 게이트 퍼셉트론 함수를 생성하시오!

```
print(OR(0,0)) #0
```

```
print(OR(1,0)) #1
```

```
print(OR(0,1)) #1
```

```
print(OR(1,1)) #1
```

답 :

```
def OR( x1, x2 ):
```

```
    w1, w2, theta = 0.5, 0.5, 0.4
```

```
    tmp = x1*w1 + x2 * w2 #입력값과 가중치의 곱의 총합
```

```
    if tmp <= theta: # 입력값과 가중치의 총합이 임계치를 넘지 않는다면
```

```
        return 0
```

```
    elif tmp > theta:
```

```
        return 1
```

```
print(OR(0,0))
```

```
print(OR(1,0))
```

```
print(OR(0,1))
```

```
print(OR(1,1))
```

### 문제20. 아까 만들었던 AND 게이트의 데이터 행렬에서 X 행렬의 1행 1열의 데이터를 가져오시오!

```
X = np.array( [[0,0], [0,1], [1,0], [1,1]] ) #입력데이터
y = np.array( [ [0], [0], [0], [1] ] ) # 정답

print(X[0][0])
```

**문제21. 위에서 만든 퍼셉트론 함수 AND에 입력값 X의 데이터를 입력하여 아래와 같이 결과를 출력하시오!**

**결과 : 0**  
**0**  
**0**  
**1**

정답 :

```
X = np.array( [[0,0], [0,1], [1,0], [1,1]] ) #입력데이터

def AND( x1, x2 ):
    w1, w2, theta = 0.5, 0.5, 0.7
    tmp = x1*w1 + x2 * w2 #입력값과 가중치의 곱의 총합
    if tmp <= theta: # 입력값과 가중치의 총합이 임계치를 넘지 않는다면
        return 0
    elif tmp > theta:
        return 1

for i in range(0,len(X)):
    print(AND(X[i][0], X[i][1]))
```

**문제22. 아래의 표와 같은 NAND 게이트 함수를 생성하고 구현하시오!**



```
def NAND( x1, x2 ):
    w1, w2, theta = 0.5, 0.5, 0.7
    tmp = x1*w1 + x2 * w2 #입력값과 가중치의 곱의 총합
    if tmp <= theta: # 입력값과 가중치의 총합이 임계치를 넘지 않는다면
        return 1
    elif tmp > theta:
        return 0

print(NAND(0,0)) #1
print(NAND(1,0)) #1
print(NAND(0,1)) #1
print(NAND(1,1)) #0
```

## ☐ XOR 게이트 P. 54

### eXclusive OR 게이트

Exclusive ? 배타적( 자기 외에는 거부한다는 뜻 )

- 1957년 로젠블라트 퍼셉트론
- 1959년 민스키가 퍼셉트론의 XOR 게이트를 문제점을 지적했다.

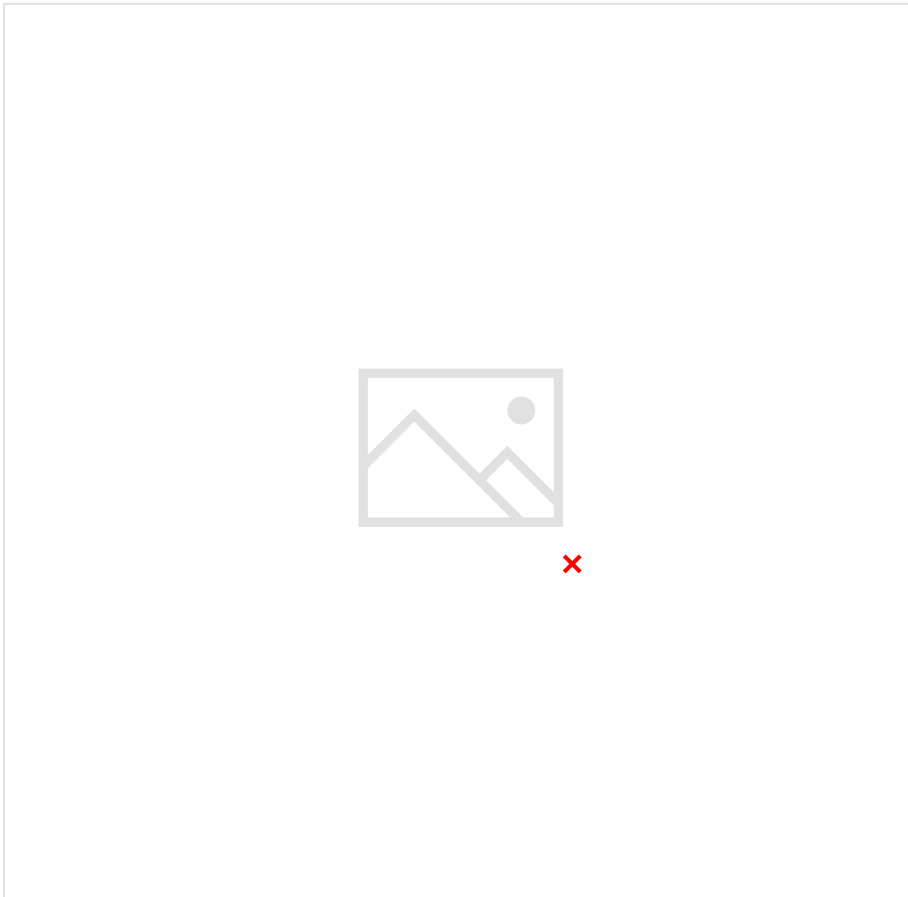
AND, OR, NAND는 단층 신경망으로도 구현이 되는데 XOR 게이트는 단층으로는 구현이 어렵다.

- 20년 후에 다층 퍼셉트론으로 비선형 영역을 분류했다.

### XOR 게이트 그림 2-5



XOR 게이트를 그래프로 시각화하기 위해 먼저 그림 2-6 OR게이트를 본다.



< 그림 2-6 >

XOR 게이트의 그래프는 그림 2-7, 2-8이다.



×

<그림 2-7>



×

<그림 2-8>

책 59페이지의 그림 2-12를 보면 다층 퍼셉트론으로 XOR 게이트를 해결한게 나와있다.



< 그림 2-12 >

**문제23. 그림 2-11에 나오는 그림으로 입력값 데이터를 받아서 XOR 게이트 결과가 출력되게 하시오!**

결과 :

```
print(XOR(0,0)) #0
print(XOR(1,0)) #1
print(XOR(0,1)) #1
print(XOR(1,1)) #0
```



**답 :**

```
def XOR( x1, x2 ):
    s1 = NAND( x1, x2 )
    s2 = OR( x1, x2 )
    y = AND( s1, s2 )
    return y
```

```
print(XOR(0,0)) #0
```

```
print(XOR(1,0)) #1
print(XOR(0,1)) #1
print(XOR(1,1)) #0
```

## ■ 가중치와 편향 구현하기 p52

1. 파라미터 : 가중치, 편향
2. 하이퍼 파라미터 : 러닝 레이트, 가중치 감소( decay ), 층수와 뉴런의 개수

**가중치 ?** 입력신호가 결과에 주는 영향력( 중요도 )를 조절하는 매개변수이다.

**편향 ?** 뉴런이 얼마나 쉽게 활성화( 결과로 1을 출력 ) 하느냐를 조정하는 매개변수이다.

예 : OR 게이트처럼 입력신호가 x1과 x2 값을 받는 경우에 편향( x0 )이 없다면 target을 분류하는 직선은 무조건 원점을 통과해야만 하기 때문에 제대로 분류할 수 없게 된다.



### 문제24. 이번에는 편향을 넣어서 AND 게이트 함수를 생성하시오!

답 :

```
def AND( x1, x2 ):
    x = np.array( [x1, x2] )
    w = np.array( [0.5, 0.5] ) # 실제로는 이 w와 b를 컴퓨터( 기계 )가 학습해서 알아내야 함
    b = -0.7
    tmp = np.sum( x*w ) + b
    if tmp <= 0:
        return 0
    else:
        return 1
```

```
print(AND(0,0)) # 0
print(AND(1,0)) # 0
```



```
print(AND(0,1)) # 0
print(AND(1,1)) # 1
```

## 문제25. OR 게이트 함수를 생성하시오!

```
def OR( x1, x2 ):
    x = np.array( [x1, x2] )
    w = np.array( [0.5, 0.5] ) # 실제로는 이 w와 b를 컴퓨터( 기계 )가 학습해서 알아내야 함
    b = -0.4
    tmp = np.sum( x*w ) + b
    if tmp <= 0:
        return 0
    else:
        return 1

print(OR(0,0)) # 0
print(OR(1,0)) # 1
print(OR(0,1)) # 1
print(OR(1,1)) # 1
```

## 문제26. ( 오늘의 마지막 문제 ) NAND 게이트 함수를 만들고 XOR 게이트 함수를 만들어서 아래의 결과가 출력되게 하시오!

```
print(XOR(0,0)) # 0
print(XOR(1,0)) # 1
print(XOR(0,1)) # 1
print(XOR(1,1)) # 0
```

답 :

```
def NAND( x1, x2 ):
    x = np.array( [x1, x2] )
    w = np.array( [0.5, 0.5] )
    b = -0.7
    tmp = np.sum( x*w ) + b
    if tmp <= 0:
        return 1
    else:
        return 0
```

```
print(NAND(0,0)) # 1
print(NAND(1,0)) # 1
print(NAND(0,1)) # 1
print(NAND(1,1)) # 0
```

```
def XOR( x1, x2 ):
    s1 = NAND(x1, x2)
```

```
s2 = OR( x1, x2 )  
y = AND(s1, s2)  
return y
```

```
print(XOR(0,0)) #0  
print(XOR(1,0)) #1  
print(XOR(0,1)) #1  
print(XOR(1,1)) #0
```

# 03/05

2021년 3월 5일 금요일 오전 9:49

## ■ 어제 배웠던 내용 복습

- 1장의 내용은 신경망 내부에서 행렬 연산을 편하게 하기 위해 사용되는 모듈인 numpy 모듈을 배웠다.
- 2장의 퍼셉트론 구현하기 위해서 함수 4개를 생성 ( AND, OR, NAND, XOR )

### 1. 단층 신경망

입력층 -----> 출력층

0층

1층

구현 가능한 게이트 : AND, OR, NAND

### 2. 다층 신경망

입력층 -----> 은닉층 -----> 출력층

구현 가능한 게이트 : XOR

## ■ 기계가 직접 학습해서 가중치(w)와 바이어스(b)를 알아내게끔 하는 방법

예제1. 아래의 두 행렬을 생성하시오!

x			target	w
x0	x1	x2		
-1	0	0	0	0.3
-1	1	0	0	0.4
-1	0	1	0	0.1
-1	1	1	1	
4x3				3x1

```
import numpy as np
```

```
x = np.array([[-1,0,0], [-1,1,0],[-1,0,1],[-1,1,1]])
```

```
w = np.array([[0.3],[0.4], [0.1]])
```

```
print(x.shape) #(4, 3)
```

```
print(w.shape) #(3, 1)
```

```
print(x.ndim) #2차원
```

print(w.ndim) #2차원

예제2. 손으로 퍼셉트론을 구현해봅시다.

	x			target	w
	x0	x1	x2		
1	-1	0	0	0	0.3
2	-1	1	0	0	0.4
3	-1	0	1	0	0.1
4	-1	1	1	1	

Diagram of a perceptron with inputs  $x_0, x_1, x_2$  and weights  $w_0, w_1, w_2$  connected to a summation node  $K$ . The output is  $f(K)$ , where  $f$  is the activation function.

$f(K) = \begin{cases} 1 & K > 0 \\ 0 & K \leq 0 \end{cases}$

Equation:  $x_0 \cdot w_0 + x_1 \cdot w_1 + x_2 \cdot w_2 = K$

Equation:  $w_i = w_i + \gamma \cdot x_i \cdot (t - f(K))$

Learning Rate (0.05)

입력신호 1 :

$$-1 \times 0.3 + 0 \times 0.4 + 0 \times 0.1 = -0.3$$

$$f(-0.3) = 0$$

입력신호 2 :

$$-1 \times 0.3 + 1 \times 0.4 + 0 \times 0.1 = 0.1$$

$$f(0.1) = 1$$

$$w_0 = 0.3 + 0.05 = 0.35$$

$$w_1 = w_1 + 0.05 \cdot x_1 \cdot x$$

$$w_1 = 0.4 + 0.05 \cdot 1 \cdot x = 0.35$$

$$w_2 = w_2 + 0.05 \cdot x_2 \cdot x$$

$$w_2 = 0.1 + 0.05 \cdot 0 \cdot x = 0.1$$

새로 갱신된 w :

$$w_0 = 0.35$$

$$w_1 = 0.35$$

$$w_2 = 0.1$$

문제27. 어제 문제 24번 AND 게이트 함수를 가져와서 위에서 구한 가중치와 바이어스를 대입해서 TARGET( 정답 )을 잘 예측하는지 테스트 하시오!

def AND( x1, x2 ):

    x = np.array( [x1, x2] )

    w = np.array( [0.35, 0.1] ) # 실제로는 이 w와 b를 컴퓨터( 기계 )가 학습해서 알아내야 함

    b = -0.35

```

tmp = np.sum( x*w ) + b
if tmp <= 0:
    return 0
else:
    return 1

```

```

print(AND(0,0)) # 0
print(AND(1,0)) # 0
print(AND(0,1)) # 0
print(AND(1,1)) # 1

```

## ■ and 게이트 데이터는 어떻게 분류가 되는 것인가?

x			target	w
x0	x1	x2		
-1	0	0	0	0.3
-1	1	0	0	0.4
-1	0	1	0	0.1
-1	1	1	1	

4x3                      3x1

b = -0.35, w1 =0.35, w2=0.1

답 :

```

import numpy as np

x = np.array([[-1,0,0], [-1,1,0],[-1,0,1],[-1,1,1]])
w = np.array([[0.3],[0.4], [0.1]])
target = np.array( [ [0], [0], [0], [1] ] )

print(w.T) #[[0.3 0.4 0.1]]
print(w.T.shape) # (1, 3)

for i in range(len(x)):
    print(np.sum(x[i]*w.T))

```

## 예제2. 입력값과 가중치의 곱의 합을 구현하시오!

```

#print(np.sum(x[0]*w.T))
#print(np.sum(x[1]*w.T))
#print(np.sum(x[2]*w.T))
#print(np.sum(x[3]*w.T))

```

또 다른 답:

```
k = x * w.T  
print(k.sum( axis = 1 ) )
```

**예제3. 입력값과 가중치의 곱의 합을 계산하는 predict 함수를 만드시오!**

```
def predict( x, w):  
    a = np.sum( x*w.T )  
    return a  
  
for i in range( len(x) ):  
    print( predict(x[i], w ) )
```

**예제4. 입력되는 값이 0보다 작거나 같으면 0을 출력하고 입력되는 값이 0보다 크면 1을 출력하는 step 함수를 생성하시오!**

```
def step_function(x):  
    if x <= 0:  
        return 0  
    else :  
        return 1  
  
print(step_function(0.3)) #1  
print(step_function(-2)) #0
```

**예제5. 위에서 출력한 k값을 방금 만든 step\_function 함수에 넣고 값을 출력하시오!**

```
def step_function(x):  
    if x <= 0:  
        return 0  
    else :  
        return 1  
  
def predict( x, w):  
    a = np.sum( x*w.T )  
    return step_function(a)  
  
for i in range( len(x) ):  
    print( predict(x[i], w ) )
```

**결과 :**

```
0  
1  
0  
1
```

### 예제6. 위에서 만든 예측값과 target 값과의 차이를 구하시오!

```
def step_function(x):
    if x <= 0:
        return 0
    else :
        return 1

def predict( x, w):
    a = np.sum( x*w.T )
    return step_function(a)

target = np.array( [ [0], [0], [0], [1] ] )

for i in range( len(x) ):
    cost = target[i] - predict(x[i], w )
```

### 예제7. 위의 cost가 0이 아니면 cost를 출력하시오!

```
def step_function(x):
    if x <= 0:
        return 0
    else :
        return 1

def predict( x, w):
    a = np.sum( x*w.T )
    return step_function(a)

target = np.array( [ [0], [0], [0], [1] ] )

for i in range( len(x) ):
    cost = target[i] - predict(x[i], w )
    if cost !=0:
        print(cost)
```

함수1

함수2

함수3

실행문( 함수를 사용하는 실행문 )

### 예제8. cost가 0이 아닐때는 가중치가 갱신될 수 있게 하시오!

```
for i in range( len(x) ):
    cost = target[i] - predict(x[i], w )
```

```

    if cost !=0:
        w = w + 0.05 * x[i] * cost
    print(w)

x = np.array([[-1,0,0], [-1,1,0],[-1,0,1],[-1,1,1]])
w = np.array([[0.3],[0.4], [0.1]])
target = np.array( [ [0], [0], [0], [1] ] )

for i in range( len(x) ):
    cost = target[i] - predict(x[i], w )
    if cost !=0:
        w = w + np.array([0.05 * x[i] * cost]).T
    print(w)

```

**문제28. ( 점심시간 문제 )** 위의 코드를 수정해서 맨 마지막에 갱신된 w값만 출력되게 하시오!

```

x = np.array([[-1,0,0], [-1,1,0],[-1,0,1],[-1,1,1]])
w = np.array([[0.3],[0.4], [0.1]])
target = np.array( [ [0], [0], [0], [1] ] )

for i in range( len(x) ):
    cost = target[i] - predict(x[i], w )
    if cost !=0:
        w = w + np.array([0.05 * x[i] * cost]).T
print(w)

```

**문제29. 비용이 0이 되고 더 이상 비용이 발생하지 않았을때의 가중치가 출력되게 하시오!**

```

import numpy as np

x = np.array( [ [ -1, 0, 0 ], [ -1, 1, 0 ], [ -1, 0, 1 ], [ -1, 1, 1 ] ] ) # 4x3
w = np.array([ [0.7], [0.4], [0.1] ] )
target = np.array( [ [0], [0], [0], [1] ] )

def step_function(x):
    if x > 0:
        return 1
    else:
        return 0

def predict(x, w):
    a = np.sum( x*w.T)
    return step_function(a)

```



```

for j in range(5):
    sum1 = 0 # sum1에는 4개의 데이터를 입력할 때 발생하는 비용(에러)가 입력될 변수
    for i in range( len(x) ): #and 게이트 4개의 입력데이터를 사용하는 루프문
        cost = target[i] - predict(x[i], w) # 비용을 계산하는 코드
        if cost != 0: # 비용이 0이 아니면
            w = w + np.array( [ 0.05 * x[i] * cost ] ).T # 가중치를 갱신해라~
        elif cost==0: # 비용이 0이면
            continue # 계속 그 다음으로 넘어가서 진행해라~
        sum1 += cost # sum1에 4개의 입력데이터를 사용했을 때 발생하는 비용 입력
    print (sum1, w.T) # 보기 편하게 하려고 w를 전치시킴

```

결과 :

```

[1] [[0.65 0.45 0.15]]
[1] [[0.6 0.5 0.2]]
0 [[0.6 0.5 0.2]]
0 [[0.6 0.5 0.2]]
0 [[0.6 0.5 0.2]]

```

**문제30. 위의 코드에서 비용(cost)가 0이 되는 시점에 break해서 loop문을 종료하고 변경된 가중치가 출력되게 하시오!**

```

import numpy as np

x = np.array( [ [ -1, 0, 0 ], [ -1, 1, 0 ], [ -1, 0, 1 ], [ -1, 1, 1 ] ] ) # 4x3
w = np.array([ 0.7], [0.4], [0.1] ] )
target = np.array( [ [0], [0], [0], [1] ] )

def step_function(x):
    if x > 0:
        return 1
    else:
        return 0

def predict(x, w):
    a = np.sum( x*w.T)
    return step_function(a)

for j in range(5):
    sum1 = 0
    for i in range( len(x) ):
        cost = target[i] - predict(x[i], w)
        if cost != 0:
            w = w + np.array( [ 0.05 * x[i] * cost ] ).T

```

```

        elif cost==0:
            continue
        sum1 += cost
    if sum1 ==0:
        break

```

```
print (sum1, w.T)
```

**문제31. 위의 코드에 for j in range(5)를 무한루프문으로 변경하시오!**

```
import numpy as np
```

```

x = np.array( [ [ -1, 0, 0 ], [ -1, 1, 0 ], [ -1, 0, 1 ], [ -1, 1, 1 ] ] ) # 4x3
w = np.array([ [0.7], [0.4], [0.1] ] )
target = np.array( [ [0], [0], [0], [1] ] )

```

```

def step_function(x):
    if x > 0:
        return 1
    else:
        return 0

```

```

def predict(x, w):
    a = np.sum( x*w.T)
    return step_function(a)

```

```

while True:
    sum1 = 0
    for i in range( len(x) ):
        cost = target[i] - predict(x[i], w)
        if cost != 0:
            w = w + np.array( [ 0.05 * x[i] * cost ] ).T
        elif cost==0:
            continue
        sum1 += cost
    if sum1 ==0:
        break
print (sum1, w.T)

```

**문제32. 위의 코드를 아래와 같이 수행되면 실행되게끔 함수로 생성하시오!**  
**( 함수이름 : perceptron\_1957 )**

```

x = np.array( [ [ -1, 0, 0 ], [ -1, 1, 0 ], [ -1, 0, 1 ], [ -1, 1, 1 ] ] ) # 4x3
w = np.array([ [0.7], [0.4], [0.1] ] )
target = np.array( [ [0], [0], [0], [1] ] )

```

```

def step_function(x):
    if x > 0:

```

```

        return 1
    else:
        return 0

def predict(x, w):
    a = np.sum( x*w.T)
    return step_function(a)

def perceptron_1957(x, w, target):
    while True:
        sum1 = 0
        for i in range( len(x) ):
            cost = target[i] - predict(x[i], w)
            if cost != 0:
                w = w + np.array( [ 0.05 * x[i] * cost ] ).T
            elif cost==0:
                continue
            sum1 += cost
        if sum1 ==0:
            break
    print('최종가중치:', w.T )

perceptron_1957(x, w, target)

```

**문제33. perceptron\_1957 함수에 OR게이트 데이터와 라벨을 입력하고 가중치를 출력 하시오!**

```

x = np.array( [ [ -1, 0, 0 ], [ -1, 1, 0 ], [ -1, 0, 1 ], [ -1, 1, 1 ] ] ) # 4x3
w = np.array([ [0.7], [0.4], [0.1] ] )
target = np.array( [ [0], [1], [1], [1] ] )

```

```

def step_function(x):
    if x > 0:
        return 1
    else:
        return 0

def predict(x, w):
    a = np.sum( x*w.T)
    return step_function(a)

def perceptron_1957(x, w, target):
    while True:
        sum1 = 0
        for i in range( len(x) ):
            cost = target[i] - predict(x[i], w)
            if cost != 0:
                w = w + np.array( [ 0.05 * x[i] * cost ] ).T
            elif cost==0:

```

```

        continue
    sum1 += abs(cost) # abs는 절대값
if sum1 == 0:
    break
print('최종가중치:', w.T )

```

perceptron\_1957(x, w, target)

## ■ 2장. 요약정리

1. 퍼셉트론 ? 인간의 뇌의 뉴런 세포를 컴퓨터로 흉내낸 것
2. 퍼셉트론의 종류 ?
  - a. 단층 퍼셉트론 : and, or nand게이트
  - b. 다층 퍼셉트론: xor 게이트
3. 인공신경망에 최종적으로 산출해야할 값? 파라미터( 가중치와 바이어스 )

## ■ 3장. 신경망

- 저자가 만들어온 가중치를 셋팅해서 필기체 데이터를 인식하는 3층 신경망을 생성  
( 저자가 학습을 해서 만들어온 가중치 )
- 신경망 안에 들어가는 함수들 소개
  - a. 활성화 함수
    - i. 계단함수
    - ii. 시그모이드 함수
    - iii. 렐루함수
  - b. 출력층 함수
    - i. 항등함수( 회귀분석 )
    - ii. 소프트맥스 함수( 분류 )
  - c. 오차 함수

이 책의 특징이 전부 파이썬 날코딩으로 다 일일이 신경망을 만드는 것이다.  
그리고서 나중에 tensorflow와 pytorch를 사용해서 신경망을 구현한다.

## ■ 계단함수 p69

"숫자 0과 1을 리턴하는 함수"

입력값  $x \leq 0$  -----> 0을 리턴

입력값  $x > 0$  -----> 1을 리턴

**예제1 :**

```
def step_function(x):
    if x > 0:
        return 1
    else:
        return 0
```

```
x_data = np.array( [-1, 0, 1 ] ) # 신경망의 데이터를 numpy array형태로 구성
print( step_function(x_data) ) #에러가 난다.
print( step_function(3.0) ) #에러가 안난다.
```

설명 : 위에서 만든 step\_function은 넘파이 배열은 넣을 수 없다.

**예제2. numpy 배열을 넣을 수 있도록 step\_function 함수를 다시 생성한다.**

```
def step_function(x):
    y = x>0
    print(y)
```

```
step_function(3.0) #True
```

```
def step_function(x):
    y = x>0 # True 또는 False가 출력이 된다.
    return y.astype(np.int) # bool type을 숫자로 변환해라~ True는 1로, False는 0으로 출력
```

```
print(step_function(np.array(3.0))) #0
print(step_function(np.array(-3.0))) #1
```

```
x_data = np.array([-1, 0, 1])
print( step_function(x_data))
```

설명 : 신경망에서 흘러가는 모든 데이터는 numpy array형태의 다차원 데이터이므로 신경망내에서 쓰여질 활성화 함수도 numpy array형태의 데이터를 받아서 처리할 수 있도록 생성되어야 한다.

**문제34. 위에서 만든 step\_function을 이용해서 책 71페이지에 나온것처럼 시각화**

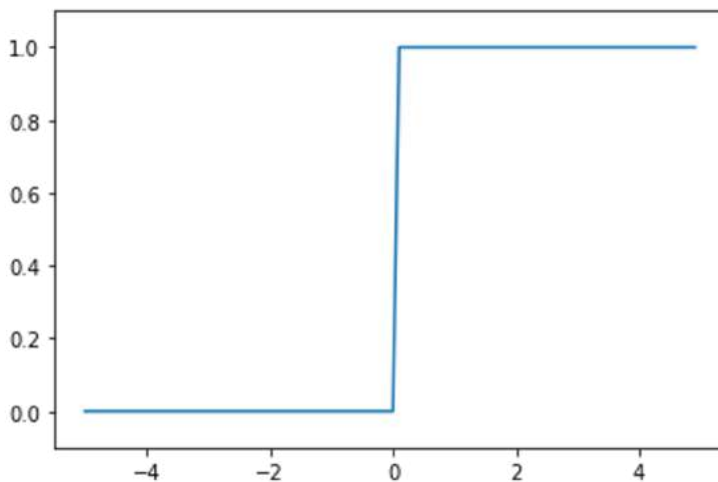
를 하시오!

```
import numpy as np
import matplotlib.pyplot as plt
```

```
def step_function(x):
    return np.array( x>0, dtype=np.int)
```

```
x = np.arange(-5.0, 5.0, 0.1) #-5부터 5까지 0.1 간격으로 숫자를 출력
y = step_function(x)
plt.plot(x,y)
plt.ylim(-0.1, 1.1)
plt.show()
```

결과 :



계단함수는 0 아니면 1이다.

문제35. 시그모이드 함수를 파이썬으로 구현하시오! ( 책 72페이지 )

$$y = \frac{1}{1 + e^{-x}}$$

시그모이드 함수식, 0~1사이의 실수를 출력

```
import numpy as np
```

```
def sigmoid(x):
    return 1/( 1 + np.exp(-x) )
```

```
x_data = np.array( [-0.5, 1.0, 2.0])
```

```
print(sigmoid(x_data))
```

**결과 :**

**[0.37754067 0.73105858 0.88079708]**

**문제36. 책 73페이지에 나오는 시그모이드 함수 그래프를 파이썬으로 그리시오!**

```
import numpy as np
```

```
def sigmoid(x):  
    return 1/( 1 + np.exp(-x) )
```

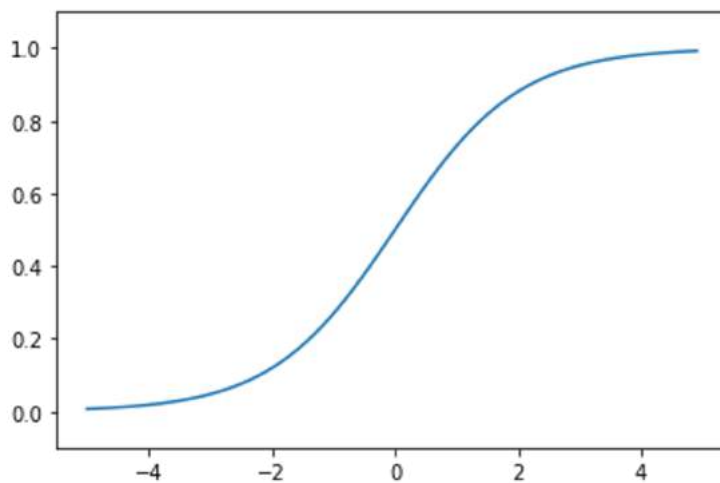
```
x = np.arange( -5.0, 5.0, 0.1)
```

```
y = sigmoid(x)
```

```
plt.plot(x,y)
```

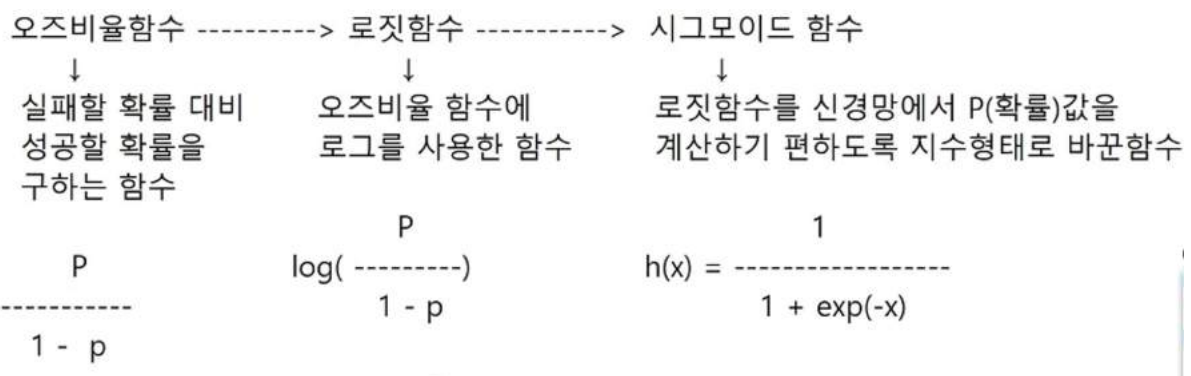
```
plt.show()
```

**결과 :**



■ **시그모이드 함수의 유래**

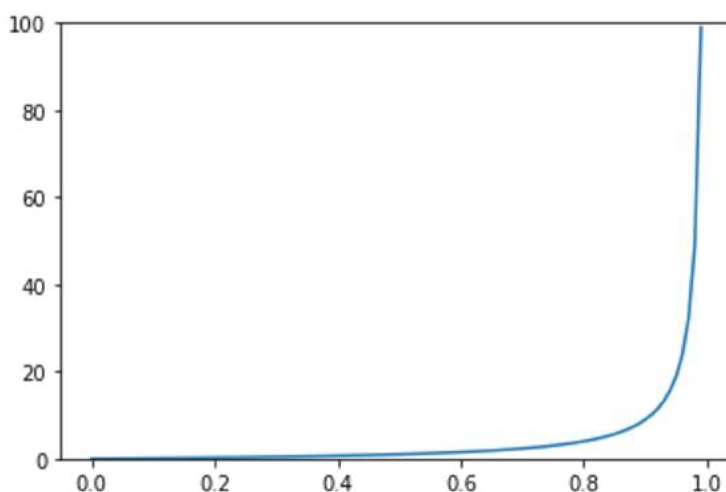
오즈함수	→ 로짓함수	→ 시그모이드 함수
$\frac{\text{성공}}{\text{실패}} = \frac{p}{1-p}$	$\log\left(\frac{p}{1-p}\right) = \log\left(\frac{p}{1-p}\right)$ $\ln\left(\frac{p}{1-p}\right) = \ln\left(\frac{p}{1-p}\right)$ $\ln\left(\frac{p}{1-p}\right) = w \cdot x + b$	$\ln\left(\frac{p}{1-p}\right) = w \cdot x + b$ $e^{\ln\left(\frac{p}{1-p}\right)} = e^{w \cdot x + b}$ $\left(\frac{p}{1-p}\right)^{\log e^e} = e^{w \cdot x + b}$ $\frac{p}{1-p} = e^{w \cdot x + b}$ $\frac{1-p}{p} = e^{-(w \cdot x + b)}$ $\frac{1}{p} - 1 = e^{-(w \cdot x + b)}$ $\frac{1}{p} = 1 + e^{-(w \cdot x + b)}$ $p = \frac{1}{1 + e^{-(w \cdot x + b)}} \quad \# w \cdot x + b = k$ $f(k) = \frac{1}{1 + e^{-k}}$



### 문제37. 오즈비율 함수 그래프를 그리시오!

```
def odds_ratio(x):
    return x / (1-x)
```

```
x = np.arange( 0, 1, 0.01)
y = odds_ratio(x)
plt.plot(x,y)
plt.ylim(0,100)
plt.show()
```

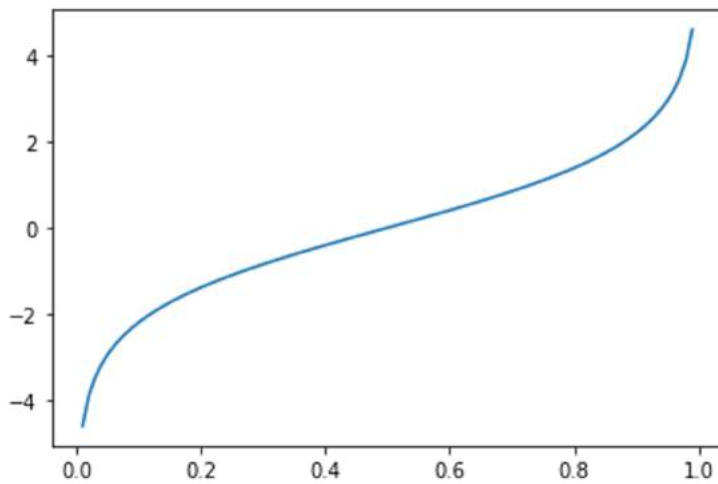




### 문제38. 로짓함수 그래프로 시각화 하시오!

```
def logit(x):  
    return np.log(x/ (1-x))
```

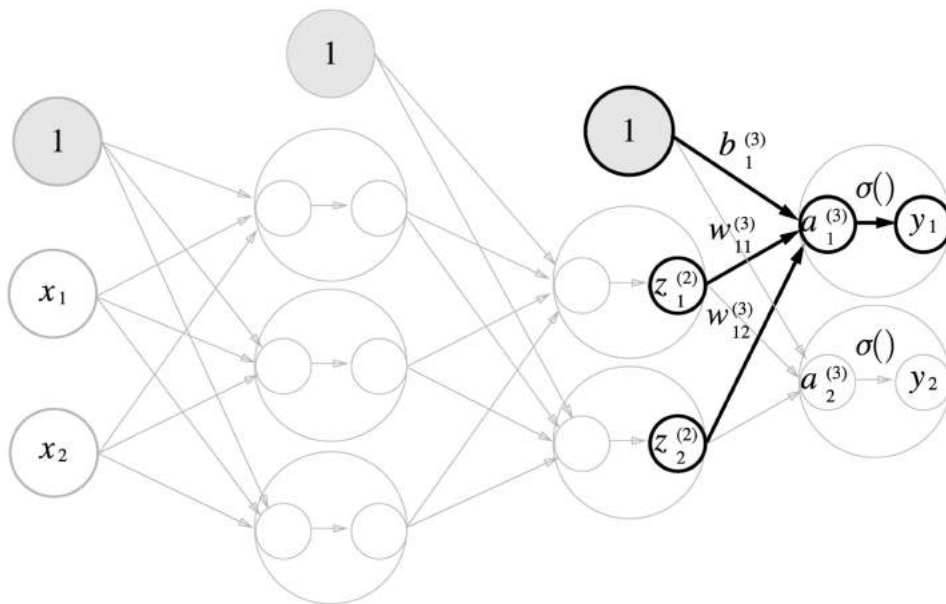
```
x = np.arange( 0.01, 1, 0.01)  
y = logit(x)  
plt.plot(x,y)  
plt.show()
```



# 03/08

2021년 3월 8일 월요일 오전 9:59

- 1장. numpy 사용법 --> 신경망 학습시 사용되는 행렬 연산에 최적화 된 모듈
- 2장. 퍼셉트론 --> 파이썬으로 퍼셉트론을 구현. 가중치가 어떻게 만들어지는지를 이해
- 3장. 3층 신경망 구성 ( 필기체 데이터 6만장 학습시킨 가중치를 저자가 만들어와서 그 가중치를 3층 신경망에 셋팅해서 필기체를 인식하게끔 함 )



※ 활성화 함수 : 신경망에 들어오는 데이터를 신경망에서 신호로 바꿔서 흘려보낼지 말지를 결정하는 함수

1. 시그모이드 함수 : 시그모이드(sigmoid)란 'S자모양'이라는 뜻이다.

```
def sigmoid(x) :  
    return 1/( 1 + np.exp( -x ) )
```

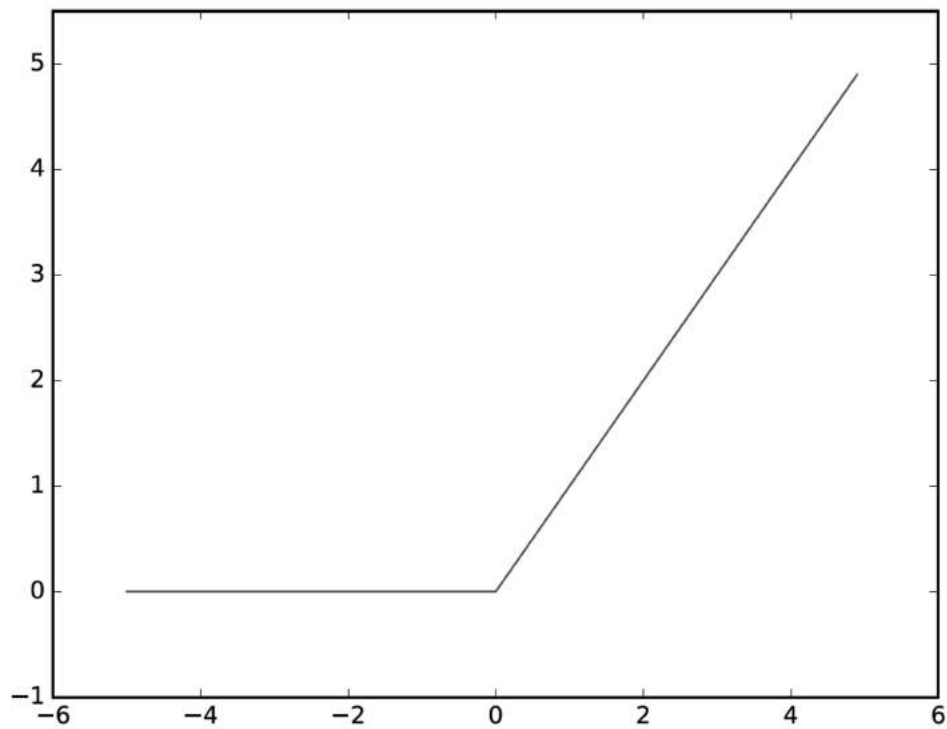
시그모이드 함수의 문제점 : 층이 깊어지면 기울기가 소실된다.

2. 렐루 함수

## ■ Relu함수 (Rectified Linear Unit / Rectified : 정류된 )

정류는 전기회로쪽 용어로 예를 들어 반파정류회로는 +/-가 반복되는 교류에서 - 흐름을 차단하는 회로이다.

Relu 함수 그래프( 그림 3-9 ) p 76처럼  $x$ 가 0 이하일때를 차단하여 아무런 값도 출력하지 않는 (0을 출력) 것이다.



### 예제1. Relu 함수를 생성하시오 !

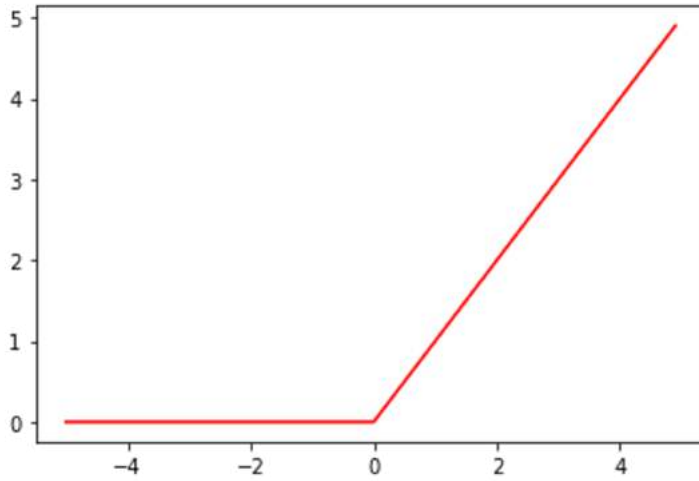
"Relu 함수는 0이하는 0으로 흘려보내고 입력이 0이 넘으면 입력값 그대로 흘려보내는 함수"

```
import numpy as np
def relu(x):
    return np.maximum( 0, x ) # 0과 x값중에 큰 값을 출력하시오!

print( relu(-2) ) #0
print( relu(0.3) ) #0.3
```

### 문제39. 위의 relu 함수를 그래프로 시각화 하시오!

```
import matplotlib.pyplot as plt
x = np.arange(-5,5,0.1)
y = relu(x)
plt.plot(x,y, color='red')
plt.show()
```



## ■ 다차원 배열계산( p77 )

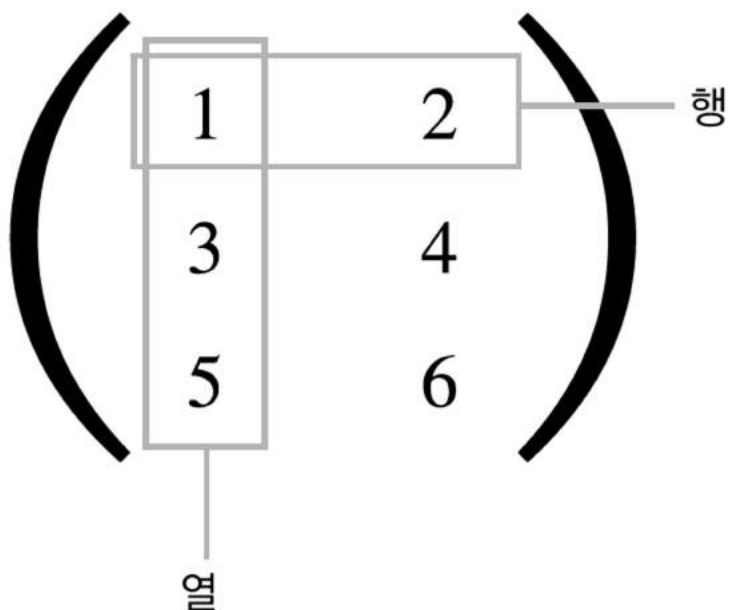
다차원 배열도 그 기본은 '숫자의 집합' 이다.

숫자를 한줄로 늘어 놓은 것이나 정사각형으로 늘어 놓은것이나 3차원으로 늘어 놓은 것 전부 다 차원 배열이라고 한다.

### 예제1. 1차원 배열을 만드는 방법

```
import numpy as np
a = np.array( [1, 2, 3, 4] )
print( np.ndim(a) ) #차원을 확인할 수 있음, 1
```

문제40. 아래의 그림의 3x2행렬을 만들고 차원을 확인하시오! ( 그림 3-10 )

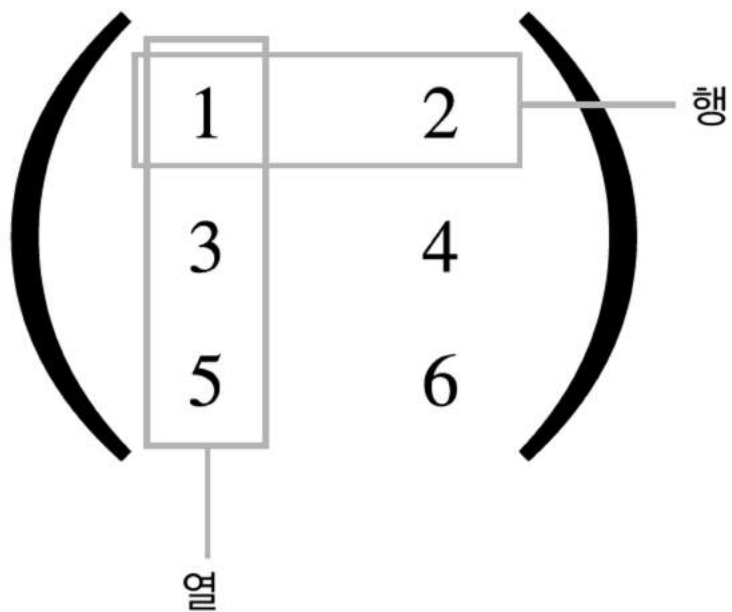


```
x = ([1,2], [3,4], [5,6])
print(np.ndim(x)) #2
```

#### 문제41. 3차원 배열을 만들어 보시오!

```
import numpy as np
b = ( [ [ [1,2], [3,4] ] , [ [5,6], [7,8] ] ] )
print(b)
print(np.ndim(b)) #3
```

#### 문제42. 아래의 2차원 배열에서 숫자 3을 출력하시오!



```
b = np.array([1,2], [3,4], [5,6])
print(b[1][0])
```

#### 문제43. 아래의 3차원 배열에서 숫자 5을 출력하시오!

```
b = ( [ [ [1,2], [3,4] ] , [ [5,6], [7,8] ] ] )
```

답 :

```
import numpy as np
b = np.array( [ [ [1,2], [3,4] ] , [ [5,6], [7,8] ] ] )

print(b[1][0][0])
```

## ■ 행렬의 내적( 행렬곱 ) p79

위의 내적을 총 3가지 방법으로 구현하시오!

### 첫번째 방법 :

```
import numpy as np

x = np.array( [ [1,2], [3,4] ] )
y = np.array( [ [5,6], [7,8] ] )
print( np.dot( x,y ) )
```

결과 :

```
[[19 22]
 [43 50]]
```

### 두번째 방법:

```
x = np.matrix( [ [1,2], [3,4] ] )
y = np.matrix( [ [5,6], [7,8] ] )
print( x*y )
```

### 세번째 방법:

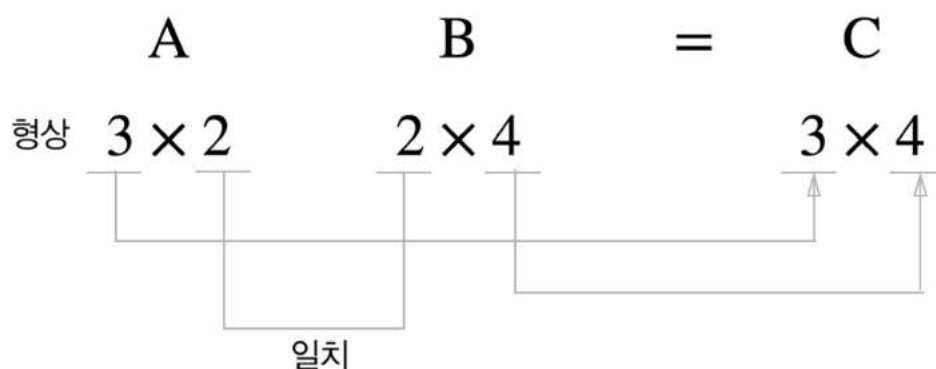
```
x = np.array( [ [1,2], [3,4] ] )
y = np.array( [ [5,6], [7,8] ] )
print( x @ y )
```

### ※ array와 matrix의 기능 차이?

array는 다차원으로 나타낼 수 있는데 matrix는 2차원 밖에 안된다.

### ※ 행렬곱( 내적 )시 주의사항!

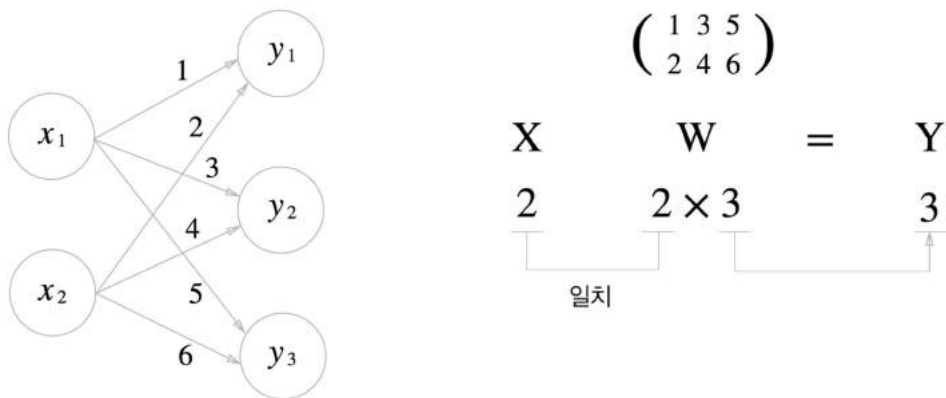
그림 3-12



다차원 배열을 곱하려면 두 행렬의 대응하는 차원의 원소 수를 일치 시켜야 한다.

## ■ 신경망 내적 ( p 82 )

그림 3-14



방정식으로 나타내면?

### ■ 신경망 내적 (p82)

$$x_1 = 1, x_2 = 2$$

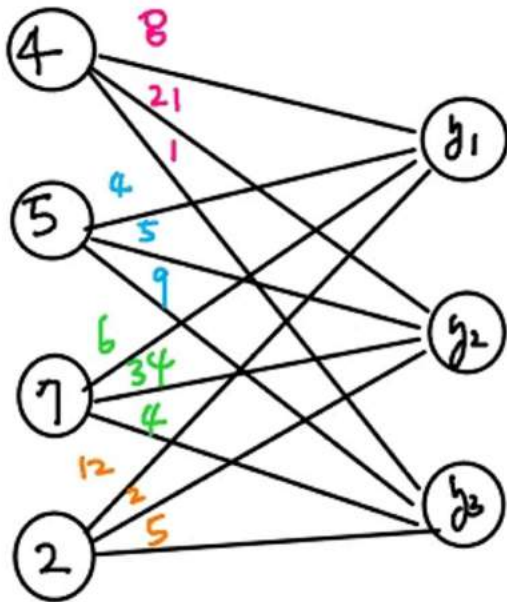
그림 3-14 를 방정식으로 나타내면 ?

$$\begin{aligned} x_1 \times 1 + x_2 \times 2 &= y_1 \\ x_1 \times 3 + x_2 \times 4 &= y_2 \\ x_1 \times 5 + x_2 \times 6 &= y_3 \end{aligned} \rightarrow (1, 2) \odot \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix} = (y_1, y_2, y_3)$$

Handwritten calculations for each output node:

- $1 \times 1 + 2 \times 2 = 5$
- $1 \times 3 + 2 \times 4 = 11$
- $1 \times 5 + 2 \times 6 = 17$

문제44. 아래의 신경망의 출력 y값을 구하시오!



import numpy as np

x1 = np.array([4,5,7,2])

x2 = np.array([[8,21,1], [4,5,9], [6,34,4], [12,2,5]])

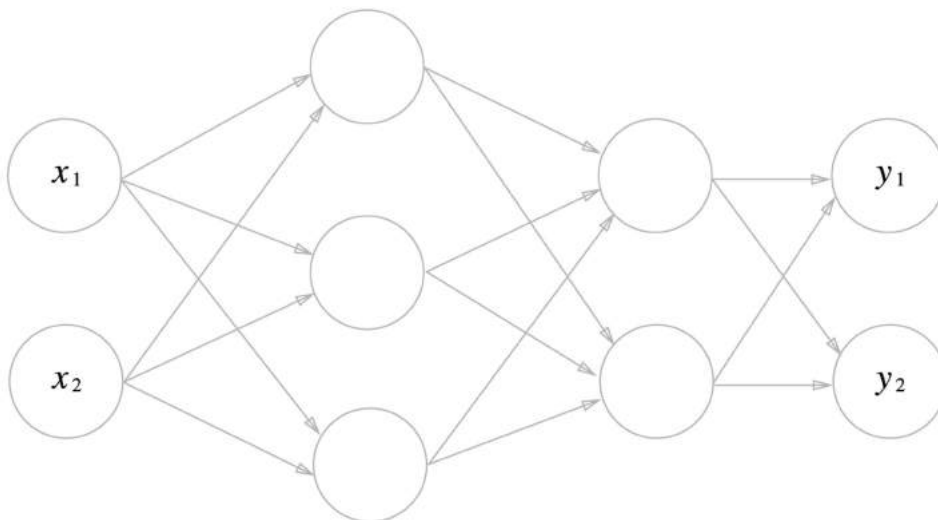
np.dot(x1,x2)

결과 :

array([118, 351, 87])

### ■ 3층 신경망 구현하기 ( p 83 )

그림 3-15



예제1. 입력층 --> 은닉 1층



```

import numpy as np

def sigmoid(x) :
    return 1/( 1 + np.exp( -x ) )

x = np.array( [1,2] )
w1 = np.array( [ [1,3,5], [2,4,6] ] )
y = np.dot( x, w1 )
y_hat = sigmoid(y)
print( y_hat ) #[0.99330715 0.9999833 0.99999996]

```

## 예제2. 입력층 --> 은닉1층 --> 은닉2층

```

def sigmoid(x) :
    return 1/( 1 + np.exp( -x ) )

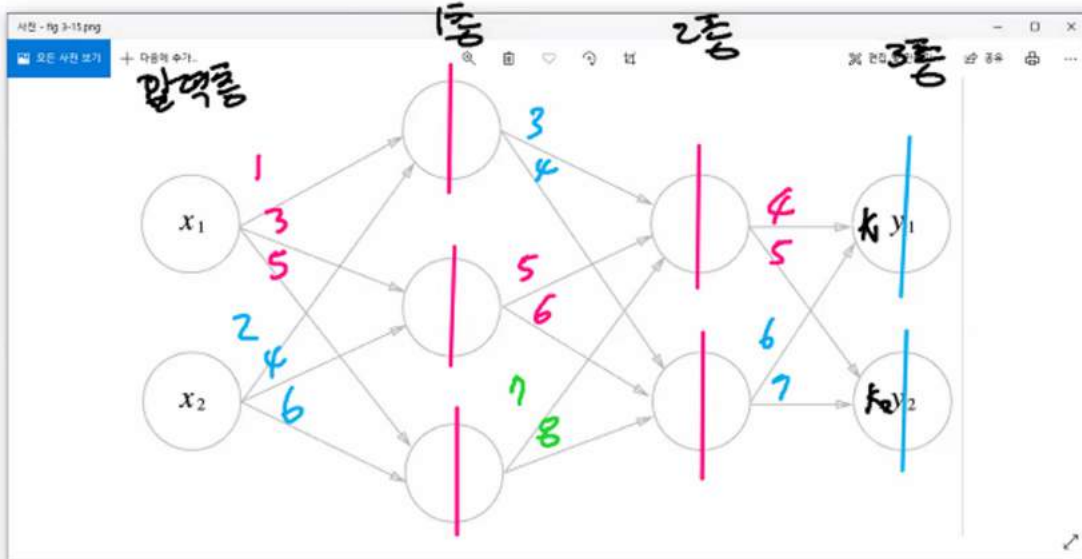
# 0층
x = np.array( [1,2] )

#1층
w1 = np.array( [ [1,3,5], [2,4,6] ] )
y = np.dot( x, w1 )
y_hat = sigmoid(y)

#2층
w2 = np.array( [[3,4], [5,6], [7,8]])
z = np.dot(y_hat, w2)
z_hat = sigmoid(z)
print(z_hat) #[0.99999969 0.99999998]

```

**문제45. ( 점심시간 문제 ) 위의 신경망을 출력층까지 구현하시오!**



```
import numpy as np
```

```
def sigmoid(x):
    return 1/(1 + np.exp(-x))
```

```
# 0층
```

```
x=np.array([1,2])
```

```
# 1층
```

```
w1 = np.array([[1,3,5],[2,4,6]])
```

```
y = np.dot(x,w1)
```

```
y_hat=sigmoid(y)
```

```
print(y_hat) # [0.99330715 0.9999833 0.99999996]
```

```
# 2층
```

```
w2 = np.array([[3,4],[5,6],[7,8]])
```

```
z = np.dot(y_hat,w2)
```

```
z_hat = sigmoid(z)
```

```
print(z_hat) # [0.99999969 0.99999998]
```

```
# 출력층
```

```
w3 = np.array([[4,5],[6,7]])
```

```
k = np.dot(z_hat,w3)
```

```
print(k) #[ 9.99999866 11.99999833]
```

## ■ 출력층 함수 ( p 90 )

" 출력층의 함수는 그동안 흘러왔던 확률들의 숫자를 취합해서 결론을 내줘야하는 함수 "

신경망으로 구현하고자 하는 문제

## 1. 회귀 ? 항등함수 ---> 입력값을 받아서 그대로 출력

예 : R을 이용한 머신러닝 때 콘크리트 강도 예측하는 머신러닝 모델을 생성  
이 때는 분류를 한게 아니라 회귀를 통한 수치를 예측 ( 콘크리트 강도 )

독립변수 : 콘크리트 재료 - 자갈 200kg, 시멘트 20포대

종속변수 : 콘크리트 강도

## 2. 분류 ? 소프트맥스 함수 ---> 책 p 91의 식 3.10

입력값을 받아서 확률벡터로 출력하는 함수

Ex ) 정상 폐사진 vs 폐결절 사진, 수화 동작을 글로 출력하는 신경망 만들기

## ■ 출력층 함수인 소프트맥스 함수 생성( p 91 )

$$y_k = \frac{\exp(a_k)}{\sum_{i=1}^n \exp(a_i)}$$

식 3-10

위의 식을 파이썬으로 그대로 만들면 에러가 나서 구현이 안된다. 왜냐하면 지수함수는 쉽게 아주 큰 값을 출력하기 때문에 컴퓨터는 큰 값을 출력이 되면 overflow가 출력 되면서 에러가 난다. 위의 수학식을 컴퓨터로 구현할 수 있게 하려면 아래와 같이 전해줘야 한다.

$$\begin{aligned} y_k &= \frac{\exp(a_k)}{\sum_{i=1}^n \exp(a_i)} = \frac{C \exp(a_k)}{C \sum_{i=1}^n \exp(a_i)} \\ &= \frac{\exp(a_k + \log C)}{\sum_{i=1}^n \exp(a_i + \log C)} \\ &= \frac{\exp(a_k + C')}{\sum_{i=1}^n \exp(a_i + C')} \end{aligned}$$

소프트 맥스 함수의 자연상수의 지수함수는 아주 큰 값을 출력한다.

자연상수 e의 10승은 20000이 넘고 e의 100승은 숫자 40개가 넘는다. e의 1000승은 무한대를 뜻하는 inf가 출력된다. 그래서 계산을 할 수가 없다.

**예:**

```
import numpy as np
print( np.exp(10) ) # 22026.465794806718
print( np.exp(100) )# 2.6881171418161356e+43
print(np.exp(1000)) #inf
```

이를 해결하기 위해서 책 93페이지의 설명처럼 입력신호중에 최댓값을 이용해서 입력 신호 값들의 최댓값으로 각각의 요소의 값을 빼준다.

**예:**

```
import numpy as np
a = np.array( [1010, 1000, 990 ] )
print( np.exp(a) )
```

결과 :

```
[inf inf inf]
```

# 확률을 구하는거라서 결과는 차이가 없다.

```
import numpy as np
a = np.array( [1010, 1000, 990 ] )
C = np.max(a)
minus = a-C
print(minus) # [0 -10 -20]
```

## 위의 코드를 이용해서 소프트맥스 함수 구현하기

식 3.11

$$\begin{aligned}
 y_k &= \frac{\exp(a_k)}{\sum_{i=1}^n \exp(a_i)} = \frac{C \exp(a_k)}{C \sum_{i=1}^n \exp(a_i)} \\
 &= \frac{\exp(a_k + \log C)}{\sum_{i=1}^n \exp(a_i + \log C)} \\
 &= \frac{\exp(a_k + C')}{\sum_{i=1}^n \exp(a_i + C')}
 \end{aligned}$$

파이썬으로 구현 :

```
a = np.array( [1010, 1000, 990 ] )
```

#### # 분자 구현

```
def softmax( a ):
    C = np.max(a)
    minus = a - C
    np_exp = np.exp(minus)
    return np_exp
```

```
softmax(a)
```

결과 :

```
array([1.00000000e+00, 4.53999298e-05, 2.06115362e-09])
```

#### # 분모까지 포함해서 구현 :

```
a = np.array( [1010, 1000, 990 ] )
```

```
def softmax( a ):
    C = np.max(a)
    minus = a - C
    exp_a = np.exp(minus)
    sum_exp_a = np.sum(exp_a)
    y = exp_a / sum_exp_a
    return y
```

```
print(softmax(a)) #[9.99954600e-01 4.53978686e-05 2.06106005e-09]
```

Ex : 정상 폐

폐결절

폐암

**문제46. 아래의 리스트의 요소를 다 더하면 숫자가 1인지 출력하시오!**

[9.99954600e-01 4.53978686e-05 2.06106005e-09]

Ex : 정상 폐

폐결절

폐암

```
a = np.array( [1010, 1000, 990 ] )
```

```
def softmax( a ):
    C = np.max(a)
    minus = a - C
    exp_a = np.exp(minus)
    sum_exp_a = np.sum(exp_a)
    y = exp_a / sum_exp_a
    return y
```

```
print( np.sum( softmax(a) ) ) #1.0
```

**문제47. 아래의 요소 3개중에 어떻게 가장 큰 값인지 인덱스 번호로 출력하시오!**

[9.99954600e-01 4.53978686e-05 2.06106005e-09]

```
a = np.array( [1010, 1000, 990 ] )
```

```
def softmax( a ):
    C = np.max(a)
    minus = a - C
    exp_a = np.exp(minus)
    sum_exp_a = np.sum(exp_a)
    y = exp_a / sum_exp_a
    return y
```

```
print(np.argmax(softmax(a))) #0
```

※ np.argmax란 ?

numpy 리스트의 요소 중 가장 큰 값의 인덱스 번호를 출력

**문제48. 지금 방금 만든 소프트맥스 함수를 점심시간에 만든 3층 신경망 맨 끝의 출력층에 k값을 입력받게끔 구현하시오! 출력은 k\_hat으로 하세요.**

```
import numpy as np
```

```
# 신경망 함수들
```

```
def sigmoid(x):
```

```
return 1/(1 + np.exp(-x))
```

```
def softmax( a ):
    C = np.max(a)
    minus = a - C
    exp_a = np.exp(minus)
    sum_exp_a = np.sum(exp_a)
    y = exp_a / sum_exp_a
    return y
```

**# 0층**

```
x=np.array([1,2])
```

**# 1층**

```
w1 = np.array([[1,3,5],[2,4,6]])
y = np.dot(x,w1)
y_hat=sigmoid(y)
print(y_hat) # [0.99330715 0.9999833 0.99999996]
```

**# 2층**

```
w2 = np.array([[3,4],[5,6],[7,8]])
z = np.dot(y_hat,w2)
z_hat = sigmoid(z)
print(z_hat) # [0.99999969 0.99999998]
```

**# 3층**

```
w3 = np.array([[4,5],[6,7]])
k = np.dot(z_hat,w3)
k_hat = softmax(k)
print( k_hat )
```

**문제49. 위의 3층 신경망 코드에서 w1, w2, w3 가중치를 하나로 모아서 심플한 코드로 작성하시오!**

" 딕셔너리를 활용하면 된다 "

- 파이썬의 자료형 5가지
- 1. 문자형
- 2. 숫자형
- 3. 리스트형
- 4. 딕셔너리형 : 키와 값으로 구성되어 있는 자료구조
- 5. 튜플

```
import numpy as np
```

```
def init_network():
```

```

network = {} # 비어있는 딕셔너리 생성
network['W1'] = np.array( [ [1, 3, 5], [2, 4, 6] ] )
network['W2'] = np.array( [ [3,4], [5, 6], [7, 8] ] )
network['W3'] = np.array( [ [4, 5], [6, 7] ] )
return network

# 가중치값을 불러온다 .
network = init_network()
w1, w2, w3 = network['W1'], network['W2'], network['W3']

# 신경망 함수들
def sigmoid(x):
    return 1 / (1 + np.exp(-x) )

def softmax(a):
    C = np.max(a)
    minus = a - C
    exp_a = np.exp(minus)
    sum_exp_a = np.sum(exp_a)
    y = exp_a / sum_exp_a
    return y

# 0층
x = np.array( [1,2] )

# 1층
y = np.dot(x,w1)
y_hat = sigmoid(y)

# 2층
z = np.dot(y_hat, w2)
z_hat = sigmoid(z)

# 3층
k = np.dot(z_hat, w3)
k_hat = softmax(k)
print(k_hat) # [0.11920296 0.88079704]

```

**문제50. 위의 sigmoid, softmax, init\_network 함수코드를 common.py라는 이름으로 메모장에 저장하고 파이썬의 워킹디렉토리에 저장하시오!**

```

import numpy as np

def init_network():
    network = {} # 비어있는 딕셔너리 생성
    network['W1'] = np.array( [ [1, 3, 5], [2, 4, 6] ] )
    network['W2'] = np.array( [ [3,4], [5, 6], [7, 8] ] )

```



```
network['W3'] = np.array( [ [4, 5], [6, 7] ] )  
return network
```

# 신경망 함수들

```
def sigmoid(x):  
    return 1 / (1 + np.exp(-x))
```

```
def softmax(a):  
    C = np.max(a)  
    minus = a - C  
    exp_a = np.exp(minus)  
    sum_exp_a = np.sum(exp_a)  
    y = exp_a / sum_exp_a  
    return y
```

※ 설명 : 위와 같이 코드를 작성하게 되면 **common.py**에 계속 코드를 추가하면 된다. 함수를 생성하면 **common.py**에 추가하면 된다.

## ■ 필기체 데이터를 신경망에 로드하기 p 96

\* 필요한 파일 2가지?

1. 필기체 데이터 ( dataset.zip )
2. 저자가 이미 만들어 놓은 가중치와 바이어스 ( sample\_weight.pkl ) <-- 피클파일

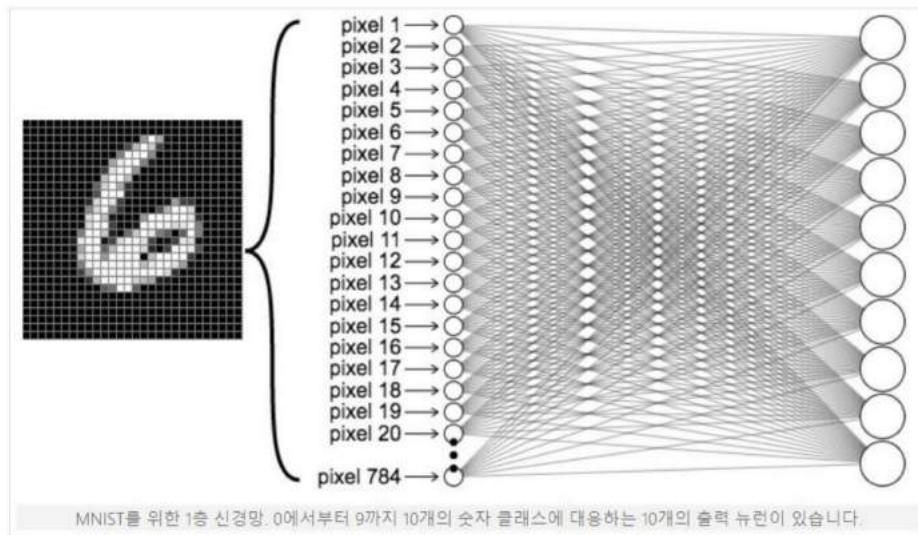
설명 : mnist 데이터를 숫자 0~9까지의 숫자 이미지로 구성되어있고 훈련 데이터가 6만장, 테스트 데이터가 1만장으로 준비되어 있다.

그림 3-24



mnist 이미지 데이터를 28x28 크기의 회색조 이미지( 1채널 )이며, 각 픽셀은 0~255까지의 값을 취한다.

워킹 디렉토리에 common이라는 패키지( 폴더 )가 있는지 먼저 찾고 없으면 common.py가 있는지 찾는다.



- **mnist 데이터를 파이썬으로 로드하기**

1. 저자가 제공하고 있는 **dataset** 폴더를 주피터의 워킹 디렉토리로 복사한다.
2. 주피터 노트북에서 아래와 같이 코드를 작성한다.

```
import sys, os
sys.path.append( os.pardir )
from dataset.mnist import load_mnist
```

```
( x_train, t_train ), ( x_test, t_test ) = load_mnist( flatten=True, normalize=False )
x_train : 훈련데이터
t_train : 훈련데이터의 정답
x_test : 테스트 데이터
t_test : 테스트 데이터의 정답
print( x_train.shape ) # (60000, 784)
```

**설명 :** **normalize :** 이미지의 픽셀값을 0.0 ~ 1.0 사이의 값으로 정규화 할지를 정한다.  
**flatten :** 입력 이미지를 1차원 배열로 만들지를 정한다.  
**load\_mnist :** 필기체 데이터를 불러오는 함수 ( 책의 저자가 만든 함수 )

**문제51. 테스트 데이터는 몇장이 있는지 확인하시오!**

```
print( x_test.shape ) # (10000, 784)
```

**문제52. 훈련 데이터의 첫번째 필기체의 숫자가 무엇인지 정답을 출력해서 알아**

**내시오.**

```
print( t_train[0] ) # 5
```

**문제53. 훈련 데이터의 첫번째 필기체 데이터를 2차원으로 해서 시각화 하시오!**

아나콘다 프롬프트 창을 열고 pillow 모듈을 설치하세요!

```
pip install pillow
```

답 :

```
from PIL import Image # 이미지를 시각화 하기 위한 모듈
```

```
( x_train, t_train ), ( x_test, t_test ) = load_mnist( flatten=True, normalize=False )
```

```
img = x_train[0] # (1,28,28) 3차원이 보임
```

```
img = img.reshape(28,28) #그래서 2차원으로 reshape 해줘야 함
```

```
def img_show( img ):
    pil_img = Image. fromarray( np.uint8(img) )
    pil_img.show()
```

```
img_show( img )
```

**문제54. mnist 데이터를 flatten 시키지 말고 출력하시오!**

```
( x_train, t_train ), ( x_test, t_test ) = load_mnist( flatten=False, normalize=False )
print(x_train.shape) #(60000, 1, 28, 28)
```

60000 : 필기체 전체 장수

1 : 색조, 1 = 흑백사진, 3 = 칼라사진

첫번째 28 : 가로 , 두번째 28 : 세로

**문제55. 아이린 사진을 파이썬에서 시각화 하시오!**

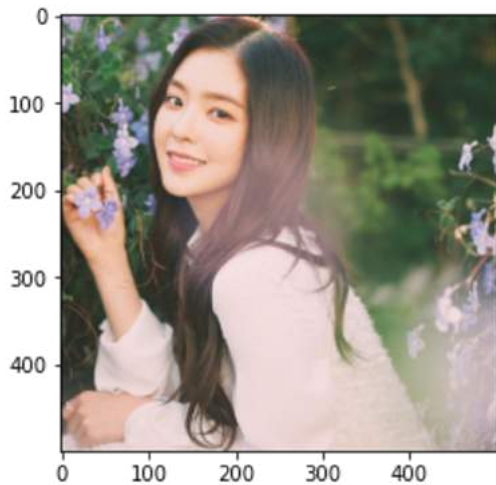
```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
```

```
img = Image.open('C:\Users\stuu03\아이린.jpg')
```

```
img_pixel = np.array(img) # 이미지를 numpy array로 변환합니다.
```

```
plt.imshow(img_pixel) # 이미지 시각화
```

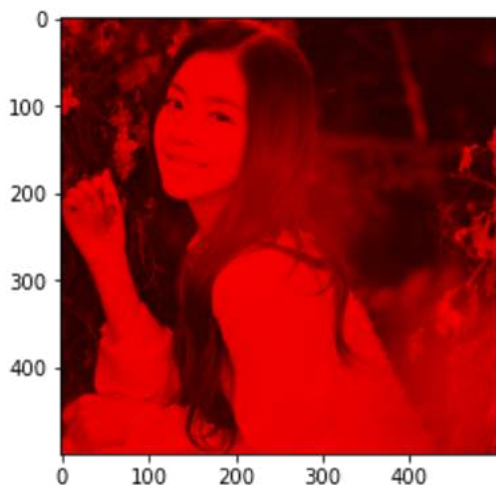
```
print(img_pixel.shape()) #(500,500,3)
```



**문제56. 아이린 사진에서 red 부분의 행렬을 취하고 red부분만 이미지로 시각화 하시오!**

```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt

img = Image.open('C:\Users\st03\아이린.jpg')
img_pixel = np.array(img) # 이미지를 numpy array로 변환합니다.
print( img_pixel[ :, :,0]) #red부분 행렬만 출력
img_pixel[:, :, 1] = 0 # green 부분을 전부 0으로 변경
img_pixel[:, :, 2] = 0 # blue부분을 전부 0으로 변경
plt.imshow(img_pixel)
plt.show()
```

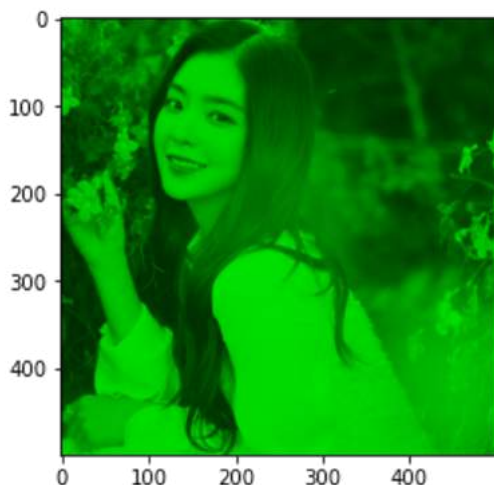


**# img\_pixel[a, b, c] a = 가로, b = 세로, c = 색조**

### 문제57 아이린 사진에서 green 부분만 시각화 하시오!

```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt

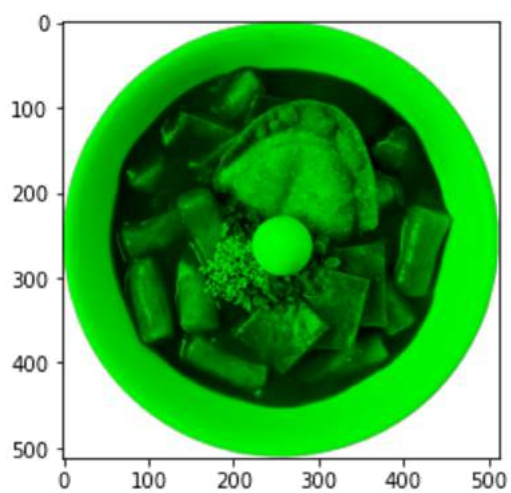
img = Image.open('C:\Users\st03\아이린.jpg')
img_pixel = np.array(img) # 이미지를 numpy array로 변환합니다.
print( img_pixel[ :, :,1]) #red부분 행렬만 출력
img_pixel[:, :, 0] = 0 # green 부분을 전부 0으로 변경
img_pixel[:, :, 2] = 0 # blue부분을 전부 0으로 변경
plt.imshow(img_pixel)
plt.show()
#print(img_pixel) #
```



### 문제58. ( 오늘의 마지막 문제 )인터넷에서 원하는 사진을 받아서 파이썬으로 시각화 하는데 red, green, blue 중 하나로만 시각화 하시오!

```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt

img = Image.open('C:\Users\st03\다이어트.jpg')
img_pixel = np.array(img) # 이미지를 numpy array로 변환합니다.
print( img_pixel[ :, :,1]) #red부분 행렬만 출력
img_pixel[:, :, 0] = 0 # green 부분을 전부 0으로 변경
img_pixel[:, :, 2] = 0 # blue부분을 전부 0으로 변경
plt.imshow(img_pixel)
plt.show()
```



# 03/09

2021년 3월 9일 화요일 오전 9:45

## 문제59. 아이린 사진을 흑백으로 변경하시오~

컬러사진 : RGB ( 3가지 색깔 )

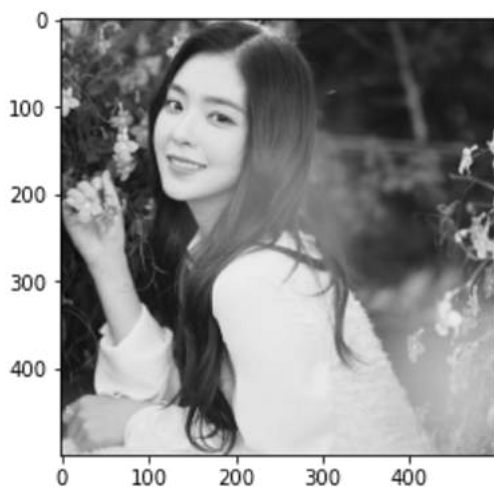
흑백사진 : Gray ( 1가지 색깔 )

```
j= 'C:\Users\WWstu03\WW아이린.jpg'
import numpy as np # 신경망에 사진을 입력할 때 숫자행렬로 입력해야 하기 때문에 필요
import matplotlib.pyplot as plt # 사진을 파이썬에서 시각화하기 위해 필요
import matplotlib.image as mpimg # 사진을 불러와서 숫자로 변환해주기 위해 필요

def rgb2gray(rgb): #흑백으로 색깔을 변경하기 위한 함수
    return np.dot(rgb[... , :3], [0.299, 0.587, 0.114])
```

**rgb[... , :3] : [행, 열, 색조]인데 이렇게 표현해 줌. 사실상 rgb[ :, :, :]와 같다.**

```
img = mpimg.imread(j)
img.shape # (500, 500, 3)
gray = rgb2gray(img)
gray.shape # (500 ,500)
plt.imshow(gray, cmap = plt.get_cmap('gray'))
plt.show()
```



#####

```
j= 'C:\Users\WWstu03\WW아이린.jpg'
import numpy as np
import matplotlib.pyplot as plt
```

```

import matplotlib.image as mpimg

def rgb2gray(rgb):
    return np.dot(rgb[...,:3], [0.299, 0.587, 0.114])

img = mpimg.imread(j)
img.shape # (500, 500, 3)

gray = rgb2gray(img)
gray.flatten() # 2차원을 1차원으로 변경

plt.imshow(gray, cmap = plt.get_cmap('gray')) #시각화 합니다.
plt.show()

```

공장 품질관리 요원( 정상품, 불량품 선별 ) --> 딥러닝 기술로 대체

**문제60. 어제 마지막 문제로 풀었던 사진을 흑백으로 변경하고 1차원으로 flatten 시키시오!**

## ■ 필기체 데이터를 인식하는 3층 신경망 구현하기

### ■ 필기체 데이터를 인식하는 3층 신경망 구현하기

**편향 ?** 뉴런이 얼마나 쉽게 활성화( 결과를 1을 출력 ) 하느냐를 조정하는 매개변수이다.  
 학습이 되면서 가중치처럼 다시 변경이 된다.

# common.py의 init\_network 함수 수정, bias 추가

```

import numpy as np

def init_network():
    network = {} # 비어있는 딕셔너리 생성
    network['W1'] = np.array( [ [1, 3, 5], [2, 4, 6] ] )
    network['W2'] = np.array( [ [3,4], [5, 6], [7, 8] ] )
    network['W3'] = np.array( [ [4, 5], [6, 7] ] )
    network['b1'] = np.array([0.1, 0.2, 0.3])
    network['b2'] = np.array([0.1, 0.2])
    network['b3'] = np.array([0.1, 0.2])
    return network

```

# 신경망 함수들



```
def sigmoid(x):
    return 1 / (1 + np.exp(-x) )

def softmax(a):
    C = np.max(a)
    minus = a - C
    exp_a = np.exp(minus)
    sum_exp_a = np.sum(exp_a)
    y = exp_a / sum_exp_a
    return y
```

## # 실제 신경망 전체코드

```
import numpy as np
from common import init_network, sigmoid, softmax
```

### # 가중치값을 불러온다 .

```
network = init_network()
w1, w2, w3 = network['W1'], network['W2'], network['W3']
b1, b2, b3 = network['b1'], network['b2'], network['b3']
```

### # 0층

```
x = np.array( [1,2] )
```

### # 1층

```
y = np.dot(x,w1) + b1
y_hat = sigmoid(y)
```

### # 2층

```
z = np.dot(y_hat, w2) + b2
z_hat = sigmoid(z)
```

### # 3층

```
k = np.dot(z_hat, w3) + b3
k_hat = softmax(k)
print(k_hat) # [0.11920296 0.88079704]
```

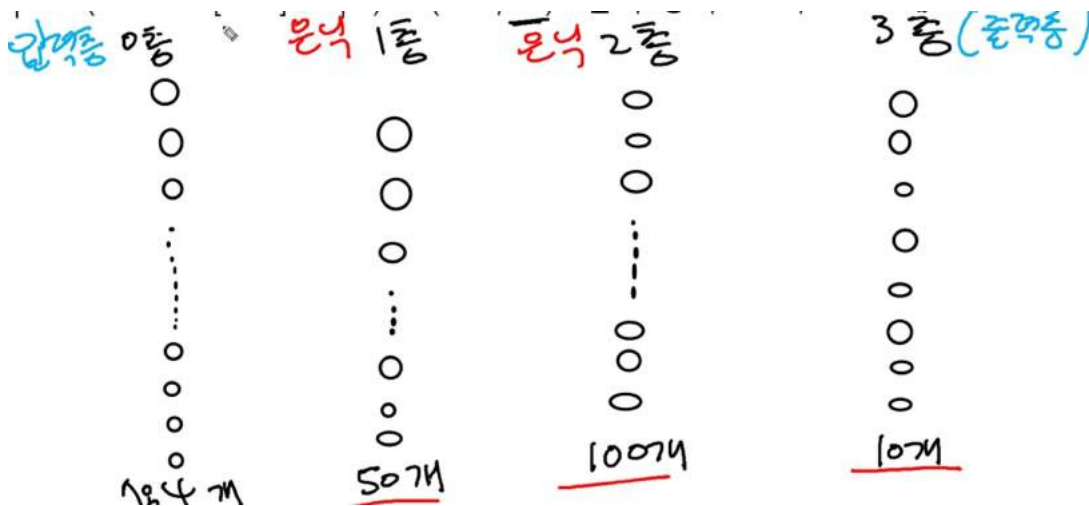


# 바이어스의 shape를 확인하는 방법

```
import pickle
from common import init_network, sigmoid, softmax

def init_network():
    with open("C:\\Users\\stuu03\\deep-learning-from-scratch-master\\ch03\\sample_weight.pkl", "rb") as f:
        network = pickle.load(f)
    return network

network = init_network()
print(network.keys()) #dict_keys(['b2', 'W1', 'b1', 'W2', 'W3', 'b3'])
print( network['W1'].shape ) # (784, 50) 은닉1층의 노드 수
```



입력층----> 은닉1층 -----> 은닉2층 ----> 출력층3층

( 100, 784 )  $\odot$  ( 784,50 ) --> ( 100, 50 )  $\odot$  ( 50, 100 ) --> ( 100, 100 )  $\odot$  ( 100, 10 )

소프트맥스함수 ? 확률벡터 출력

```
import pickle
from common import init_network, sigmoid, softmax
```

## 필기체 데이터를 불러오는 코드

```
import numpy as np
from common import init_network, sigmoid, softmax
from dataset.mnist import load_mnist
```

# 1. 데이터를 불러옵니다.

(x\_train, t\_train), (x\_test, t\_test ) = load\_mnist( flatten=True, normalize=True,

```
one_hot_label=False)
```

```
print( t_train[0:10] ) #[5 0 4 1 9 2 1 3 1 4]
```

# 설명 : one\_hot\_label은 정답을 숫자로 표현할건지 아니면 아래와 같이  
# one hot encoding 할건지 결정하는 파라미터이다.

**# 2. 가중치값을 불러옵니다. (저자가 미리 학습 시킨 가중치와 바이어스)**

```
network = init_network()  
w1, w2, w3 = network['W1'], network['W2'], network['W3']  
b1, b2, b3 = network['b1'], network['b2'], network['b3']
```

**#3. 신경망을 구성합니다.**

# 0층

```
x = x_train[0:10] #일단 10개의 필기체 데이터를 작성합니다.
```

# 1층

```
y = np.dot(x,w1) + b1  
y_hat = sigmoid(y)
```

# 2층

```
z = np.dot(y_hat, w2) + b2  
z_hat = sigmoid(z)
```

# 3층

```
k = np.dot(z_hat, w3) + b3  
k_hat = softmax(k)  
print(np.argmax(k_hat, axis=1)) # [5 0 4 1 9 2 1 3 1 4], axis=1 은 축
```

**문제61. 위의 예측한 10장중에 실제로 몇개를 맞췄는지 확인해보시오!**

**# 1. 데이터를 불러옵니다.**

```
(x_train, t_train), (x_test, t_test) = load_mnist( flatten=True, normalize=True,  
one_hot_label=False)
```

```
print( t_train[0:10] ) #[5 0 4 1 9 2 1 3 1 4]
```

**# 2. 가중치값을 불러옵니다. (저자가 미리 학습 시킨 가중치와 바이어스)**

```
network = init_network()  
w1, w2, w3 = network['W1'], network['W2'], network['W3']  
b1, b2, b3 = network['b1'], network['b2'], network['b3']
```

**#3. 신경망을 구성합니다.**

### # 0층

```
x = x_train[0:10] #일단 10개의 필기체 데이터를 작성합니다.
```

### # 1층

```
y = np.dot(x,w1) + b1  
y_hat = sigmoid(y)
```

### # 2층

```
z = np.dot(y_hat, w2) + b2  
z_hat = sigmoid(z)
```

### # 3층

```
k = np.dot(z_hat, w3) + b3  
k_hat = softmax(k)  
print(np.argmax(k_hat, axis=1)) # [5 0 4 1 9 2 1 3 1 4], axis=1 은 축, 예측값  
print( t_train[0:10] )
```

문제62. ( 점심시간 문제 ) 이번에는 훈련 데이터 총 100장을 흘려보내고 100장 중 몇개를 맞추었는지 확인하시오!

## ■ 3장의 책 100페이지에 나온 코드가 3장의 내용을 완성하는 코드

"저자가 만들어 온 가중치와 바이어스를 이용해서 신경망을 만들고 필기체 데이터를 잘 예측하는지 확인하는 코드 "

```
import numpy as np  
from common import init_network, sigmoid, softmax  
from dataset.mnist import load_mnist
```

# 파이썬 기초 --> 함수생성 --> 함수로 클래스 생성

# 1. 데이터를 불러옵니다. ( 얀르쿰 교수님이 만든 필기체 )

```
def get_data():  
    (x_train, t_train), (x_test, t_test ) = load_mnist( flatten=True, normalize=True,  
    one_hot_label=False)  
    return x_test, t_test
```

## # 2. 가중치와 바이어스 값을 불러와서 3층 신경망에 흘려보내는 함수

```
def predict( network, x ) :  
    # network = init_network()  
    w1, w2, w3 = network['W1'], network['W2'], network['W3']  
    b1, b2, b3 = network['b1'], network['b2'], network['b3']
```

### # 3. 신경망을 구성합니다.

#### # 0층

```
# x = x_train[0:100] # 일단 10개의 필기체 데이터를 구성합니다.
```

#### # 1층

```
y = np.dot(x,w1) + b1  
y_hat = sigmoid(y)
```

#### # 2층

```
z = np.dot(y_hat, w2) + b2  
z_hat = sigmoid(z)
```

#### # 3층

```
k = np.dot(z_hat, w3) + b3  
k_hat=softmax(k)  
return k_hat
```

## # 3. 위에서 만든 get\_data 함수와 predict 함수를 가져다가 실행하는 부분 ( p 101 )

```
x, t = get_data() #테스트 데이터와 테스트 데이터의 정답을 불러오는 코드  
network = init_network() # 저자가 만들어온 가중치와 바이어스를 불러오는 코드
```

```
accuracy_cnt = 0  
for i in range( len(x) ):  
    y = predict( network, x[i] )  
    p = np.argmax(y)  
    if p == t[i] :  
        accuracy_cnt +=1  
  
print( '정확도', accuracy_cnt / len(x) )
```

## 문제63. 10000장 전체를 다 돌리지 말고 1000장만 돌려서 정확도를 확인하십시오!

```
import numpy as np  
from common import init_network, sigmoid, softmax  
from dataset.mnist import load_mnist
```

# 파이썬 기초 --> 함수생성 --> 함수로 클래스 생성

# 1. 데이터를 불러옵니다. ( 안르쿤 교수님이 만든 필기체 )

```
def get_data():
    (x_train, t_train), (x_test, t_test) = load_mnist( flatten=True, normalize=True,
    one_hot_label=False)
    return x_test, t_test
```

# 2. 가중치와 바이어스 값을 불러와서 3층 신경망에 흘려보내는 함수

```
def predict( network, x ) :
    #network = init_network( )
    w1, w2, w3 = network['W1'], network['W2'], network['W3']
    b1, b2, b3 = network['b1'], network['b2'], network['b3']
```

# 3. 신경망을 구성합니다.

# 0층

# x = x\_train[0:100] # 일단 10개의 필기체 데이터를 구성합니다.

# 1층

y = np.dot(x,w1) + b1

y\_hat = sigmoid(y)

# 2층

z = np.dot(y\_hat, w2) + b2

z\_hat = sigmoid(z)

# 3층

k = np.dot(z\_hat, w3) + b3

k\_hat=softmax(k)

return k\_hat

# 3. 위에서 만든 get\_data 함수와 predict 함수를 가져다가 실행하는 부분 ( p 101 )

x, t = get\_data() #테스트 데이터와 테스트 데이터의 정답을 불러오는 코드

network = init\_network() # 저자가 만들어온 가중치와 바이어스를 불러오는 코드

accuracy\_cnt = 0

for i in range(1000):

y = predict( network, x[i] )

p = np.argmax(y)

if p == t[i] :

accuracy\_cnt +=1

print( '정확도', accuracy\_cnt / 1000 ) # 0.932

**문제64. 훈련데이터 6만장을 다 불러와서 신경망에 넣고 전체 6만개 중에 몇 개를 맞추는지 정확도를 확인하시오!**

```
import numpy as np
from common import init_network, sigmoid, softmax
from dataset.mnist import load_mnist
```

**# 파이썬 기초 --> 함수생성 --> 함수로 클래스 생성**

**# 1. 데이터를 불러옵니다. ( 얀르륵 교수님이 만든 필기체 )**

```
def get_data():
    (x_train, t_train), (x_test, t_test) = load_mnist( flatten=True, normalize=True,
    one_hot_label=False)
    return x_train, t_train
```

**# 2. 가중치와 바이어스 값을 불러와서 3층 신경망에 흘려보내는 함수**

```
def predict( network, x ) :
    #network = init_network( )
    w1, w2, w3 = network['W1'], network['W2'], network['W3']
    b1, b2, b3 = network['b1'], network['b2'], network['b3']

    # 3. 신경망을 구성합니다.
    # 0층
    # x = x_train[0:100] # 일단 10개의 필기체 데이터를 구성합니다.
    # 1층
    y = np.dot(x,w1) + b1
    y_hat = sigmoid(y)
    # 2층
    z = np.dot(y_hat, w2) + b2
    z_hat = sigmoid(z)
    # 3층
    k = np.dot(z_hat, w3) + b3
    k_hat=softmax(k)
    return k_hat
```

**# 3. 위에서 만든 get\_data 함수와 predict 함수를 가져다가 실행하는 부분 ( p 101 )**

```
x, t = get_data() #테스트 데이터와 테스트 데이터의 정답을 불러오는 코드
network = init_network() # 저자가 만들어온 가중치와 바이어스를 불러오는 코드

accuracy_cnt = 0
for i in range(len(x)):
    y = predict( network, x[i] )
    p = np.argmax(y)
    if p == t[i] :
        accuracy_cnt +=1

print( '정확도', accuracy_cnt / len(x) ) # 0.9357666666666666
```



**문제65. 위의 코드에 1번과 2번을 클래스로 생성하시오!**

**( 클래스 : three\_nn )**

```
class Three_nn():

    import numpy as np
    import pickle
    from common import init_network, sigmoid, softmax
    from dataset.mnist import load_mnist

    # 파이썬 기초 --> 함수생성 --> 함수로 클래스 생성

    # 1. 데이터를 불러옵니다. ( 얀르쿤 교수님이 만든 필기체 )
    def get_data(self):
        (x_train, t_train), (x_test, t_test ) = load_mnist( flatten=True, normalize=True,
one_hot_label=False)
        return x_train, t_train

    # 2. 가중치와 바이어스 값을 불러와서 3층 신경망에 흘려보내는 함수

    def predict(self, network, x ) :
        #network = init_network( )
        w1, w2, w3 = network['W1'], network['W2'], network['W3']
        b1, b2, b3 = network['b1'], network['b2'], network['b3']

        # 0층
        # x = x_train[0:100] # 일단 10개의 필기체 데이터를 구성합니다.

        # 1층
        y = np.dot(x,w1) + b1
        y_hat = sigmoid(y)

        # 2층
        z = np.dot(y_hat, w2) + b2
        z_hat = sigmoid(z)

        # 3층
        k = np.dot(z_hat, w3) + b3
        k_hat=softmax(k)
        return k_hat

    def init_network(self):
        with open("C:\Users\stuu03\deep-learning-from-scratch-master\ch03\
sample_weight.pkl", "rb") as f:
            network = pickle.load(f)
        return network
```

---

```
n1 = Three_nn() # 객체화 시킨다. 설계를 가지고 제품을 만든다.
```

# 3. 위에서 만든 get\_data 함수와 predict 함수를 가져다가 실행하는 부분 ( p 101 )

```
x, t = get_data() #테스트 데이터와 테스트 데이터의 정답을 불러오는 코드
```

```
network = n1.init_network() # 저자가 만들어온 가중치와 바이어스를 불러오는 코드
```

```
accuracy_cnt = 0
```

```
for i in range(len(x)):
```

```
    y = predict( network, x[i] )
```

```
    p = np.argmax(y)
```

```
    if p == t[i] :
```

```
        accuracy_cnt +=1
```

```
print( '정확도', accuracy_cnt / len(x) ) # 0.9357666666666666
```

**문제66. 훈련 데이터의 첫번째 데이터를 3층 신경망에 넣고 예측값을 출력 하시오!**

```
n1 = Three_nn()
```

# 3. 위에서 만든 get\_data 함수와 predict 함수를 가져다가 실행하는 부분 ( p 101 )

```
x, t = get_data() #테스트 데이터와 테스트 데이터의 정답을 불러오는 코드
```

```
n1.network = init_network() # 저자가 만들어온 가중치와 바이어스를 불러오는 코드
```

```
n1 = Three_nn() # 객체화 시킨다. 설계를 가지고 제품을 만든다.
```

# 3. 위에서 만든 get\_data 함수와 predict 함수를 가져다가 실행하는 부분 ( p 101 )

```
x, t = n1.get_data() #테스트 데이터와 테스트 데이터의 정답을 불러오는 코드
```

```
network = n1.init_network() # 저자가 만들어온 가중치와 바이어스를 불러오는 코드
```

```
result = n1.predict( network, x[0] )
```

```
print(np.argmax( result ) ) # 5
```

```
print( t[0] ) #5
```

**문제67. 아이린 사진을 필기체를 인식하는 3층 신경망에 넣으면 뭐가 출력 이 될까?**

( 1, 250000 ) ◎ ( 784, 50 )

**답 : 차원이 일치하지 않아서 실행되지 않는다.**

## ■ 배치처리 ( p 102 )

훈련 데이터가 6만장이나 되는데 6만장을 한번에 신경망에 넣고 학습을 시키게 되면 컴퓨터가 메모리 사용량이 초과해서 수행이 되지 않는다.

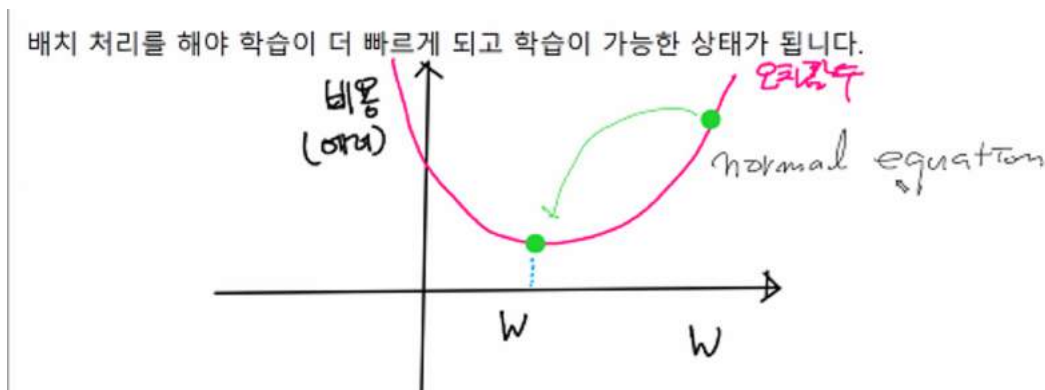
예 : 사람이 책을 볼 때도 한번에 책 한권을 동시에 볼 수 없고 한페이지씩 보듯이 컴퓨터도 마찬가지로 메모리가 허용하는 내에서 여러 페이지를 학습 할 수 있도록 해주는 것이 배치처리이다.

필기체 6만장을 학습할 때 1개씩 학습한다면 6만번 실행을 predict 해야 한다.

( predict 함수를 6만번 실행해야 한다. )

필기체 6만장을 학습할 때 한번에 100개씩 학습한다면 predict함수를 몇번 실행하면 될까?  
600번만 실행하면 된다.

배치 처리를 해야 학습이 더 빠르게 되고 컴퓨터로 학습이 가능한 상태가 된다.



**문제68. 지금은 훈련 데이터 6만장을 한번에 predict에 넣고 예측하는 것이 있는데 그러지 말고 100개씩 넣고 예측해서 정확도가 600개가 a라는 리스트에 담기게 하시오!**

```
n1 = Three_nn() # 객체화 시킨다. 설계도를 가지고 제품을 만든다.
```

```
x, t = n1.get_data() #테스트 데이터와 테스트 데이터의 정답을 불러오는 코드
```

```
network = n1.init_network() # 저자가 만들어온 가중치와 바이어스를 불러오는 코드
```

```
a = []
```

```
batch_size = 100
```

```
accuracy_cnt = 0
```

```

for i in range(0,len(x), batch_size): # 0, 100, 200, 300, ...
    y = predict( network, x[i :i+batch_size] ) # x[0:100] , x[100,200], 000
    y_hat = np.argmax(y, axis = 1) # 100개의 예측 숫자들이 출력됨
    a.append( sum( y_hat == t[ i : i + batch_size ] ) /100 ) # 예측 100개와 정답 100개를
    비교해서 정확도 계산

print(a)
print( len(a) ) # 600
print( np.mean(a) ) # 0.9357666666666665

```

## ■ 3장의 내용 총 정리

1. 신경망안에 들어가는 함수들인 시그모이드 함수, 렐루 함수, 소프트 맥스 함수, 파이썬으로 생성함
2. 저자가 만들어 놓은 가중치 피클파일로 3층 신경망을 구현
3. 배치단위로 데이터를 신경망으로 흘려보내는 이유와 구현

신경망이 학습할 때는 데이터 전체를 한번에 학습할 수는 없고 조금씩 학습해서 전체를 다 학습해라 라고 하는게 배치처리이다.

## ■ 4장. 신경망 학습( 우리가 직접 신경망의 가중치와 바이어스를 생성 )

- 신경망을 학습 시키기 위해서 알아야하는 4장의 내용 :
1. 오차함수 : 신경망이 뭘 잘못하고 있는지 깨닫게 해주는 함수
  2. 미니배치 : 학습할 때 데이터를 한꺼번에 신경망에 넣는게 아니라 조금씩( 몇 백장 ) 신경망에 넣고 학습시키는 것을 말함
  3. 수치미분 : 오차함수의 기울기를 구해서 기울기 만큼 가중치를 갱신해주는 역할을 할 때 필요한 함수

## ■ 오차함수 p 111

시그모이드 함수, 렐루함수, 소프트맥스 함수, 오차함수

예상값과 실제값과의 오차를 신경망에 역전파 시켜주기 위해서 필요한 함수

- 신경망의 잘못을 깨닫게 해주는 함수

1. 평균제곱 오차함수 ( Mean Squared Error ) : 회귀분석시 사용
2. 교차 엔트로피 오차함수 ( Cross Entropy Error ) : 분류 문제를 풀 때 사용

## ■ 평균제곱 오차 함수

책 112. 식 4.1

$$E = \frac{1}{2} \sum_k (y_k - t_k)^2$$

```
import numpy as np

y = [0.1, 0.05, 0.6, 0.0, 0.05, 0.1, 0.0, 0.1, 0.0, 0.0] # 예측 숫자
t = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0] # 정답 숫자

def mean_squared_error( y , t ):
    return 0.5 * np.sum( ( np.array(y) - np.array(t) )**2 )

print(mean_squared_error( y , t )) #0.097500000000000003
```

**문제69. 숫자 7로 예측한 결과와 정답 숫자 2와의 오차를 구하시오!**

```
y = [0.1, 0.05, 0.1, 0.0, 0.05, 0.1, 0.0, 0.6, 0.0, 0.0] # 예측 숫자 7
t = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0] # 정답 숫자 2

def mean_squared_error( y , t ):
    return 0.5 * np.sum( ( np.array(y) - np.array(t) )**2 )

print(mean_squared_error( y , t )) #0.097500000000000003
```

설명 : 위의 오차 0.59는 오차가 작아서 분류문제를 해결하기 위해서는 더 큰 오차로 응답 해줘야 한다. 그래서 필요한 함수가 바로 교차엔트로피 함수이다.

## ■ 교차 엔트로피 함수( p 114, 식 4.2 )

$$E = -\sum_k t_k \log y_k$$

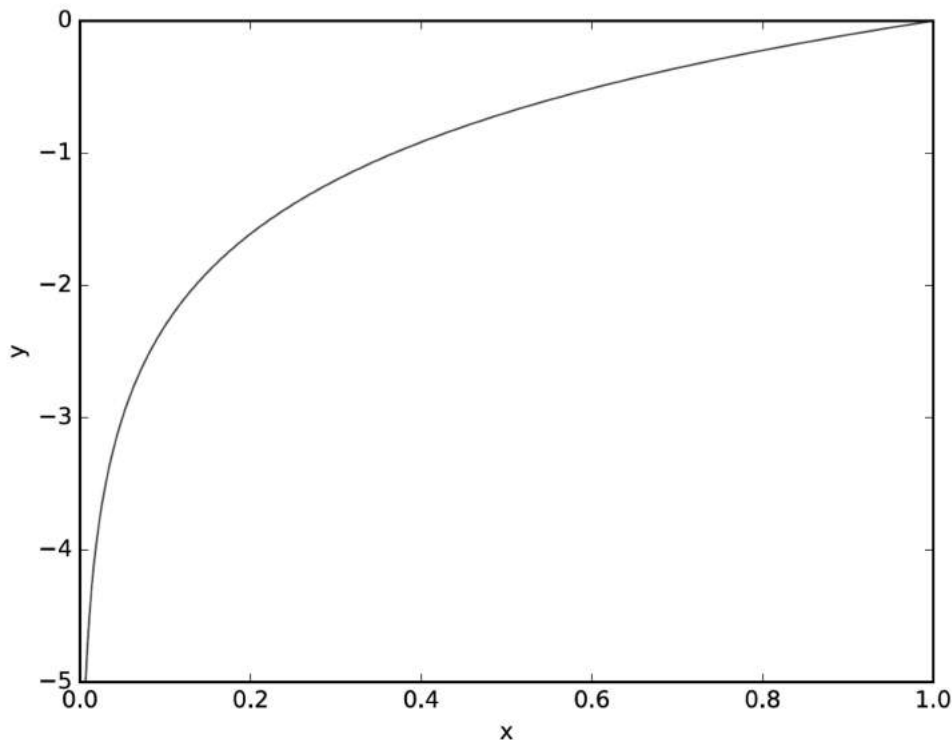
```
y = np.array( [0.1, 0.05, 0.1, 0.0, 0.05, 0.1, 0.0, 0.6, 0.0, 0.0] ) # 예측 숫자 7
t = np.array( [0, 0, 1, 0, 0, 0, 0, 0, 0, 0] ) # 정답 숫자 2

def cross_entropy_error( y, t ):
    return -np.sum( t* np.log(y) )

print( cross_entropy_error( y, t ) ) # nan
```

설명 : 결과가 왜 nan이 나왔냐면 log 함수에 숫자 0이 들어갔기 때문이다.  
log에 숫자 0이 들어가면 마이너스 무한대가 된다.

책 114페이지 그림 4-3



```
y = np.array( [0.1, 0.05, 0.1, 0.0, 0.05, 0.1, 0.0, 0.6, 0.0, 0.0] ) # 예측 숫자 7
t = np.array( [0, 0, 1, 0, 0, 0, 0, 0, 0, 0] ) # 정답 숫자 2

def cross_entropy_error( y, t ):
```

```

delta = 1e-7
return -np.sum( t* np.log(y + delta) )

print( cross_entropy_error( y, t ) ) # 2.302584092994546

```

설명 : 평균제곱 오차함수는 오차가 0.59인데 교차 엔트로피 함수는 오차가 2.3으로 훨씬 큰 오차를 출력한다. 분류문제를 풀 때는 교차 엔트로피 오차함수를 사용해야한다.

## ■ 미니 배치 학습 p 115

" 훈련 데이터 중에 일부만 골라서 학습하는 방법 "

" 표본을 뽑아서 학습 시킨다 "

3장에서는 저자가 만들어 온 가중치를 인공신경망에 셋팅해서 필기체 분류 신경망을 구현을 했다면 4장은 우리가 직접 신경망을 학습 시킬것이다.

- 6만장을 한번에 신경망에 넣으면 컴퓨터가 터져버린다.
- 6만장 중 한장씩 신경망에 입력 : 시간이 많이 걸림
- 6만장 중 100장씩 신경망에 입력 : 학습시간이 빨라짐

( 100, 784 ) ⊗ ( 784, 50 ) --> ( 100, 50 ) ⊗ ( 50, 100 ) --> ( 100, 100 ) ⊗ ( 100, 10 )

100장을 추출할 때 복원추출, 비복원 추출 할것인가는 크게 중요하지 않고 여러 번 반복해서 학습하다보면 학습이 된다.

100장씩 600번을 훈련시키면 그것이 1 epoch이다.

60000장 : 100장 x 600번

**문제70. 0 ~ 60000의 숫자 중에서 무작위로 10개를 출력하시오!**

```

import numpy as np
print( np.random.choice( np.arange( 0, 60001 ), 10 ) )

```

**문제71. 0 ~ 60000의 숫자 중에서 무작위로 100개를 출력하시오!**

```

import numpy as np

```

```
print( np.random.choice( np.arange( 0, 60001 ), 100 ) )
```

**문제72.** 3장에서 마지막으로 작성한 필기체 숫자 예측하는 전체코드를 가져와서 100개를 랜덤추출하여 예측하게 할 수 있도록 코드를 수정하시오!

```
n1 = Three_nn()

x, t = n1.get_data() # 테스트 데이터와 테스트 데이터의 정답을 불러오는 코드
network = n1.init_network() # 저자가 만들어온 가중치와 바이어스를 불러오는 코드
a = []
batch_size = 100
accuracy_cnt = 0

for i in range( 0, len(x), batch_size ):
    batch_mask = np.random.choice( 60000, 100 ) #100개씩 복원추출
    y = predict( network, x[batch_mask] )
    y_hat = np.argmax(y, axis=1)
    a.append( sum(y_hat == t[batch_mask]) / 100 )

print ( len(a) )
print ( np.mean(a) )
```

## ■ 수치미분 : p 121

**신경망을 학습 시킬 때 왜 미분이 필요한가 ?**

답 : 가중치를 갱신해주기 위해서  
가중치 = 가중치 - 기울기

**책 p 121 수식 4.4**

$$\frac{df(x)}{dx} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

```
def numerical_diff( f, x ) :
    h = 10e-50 # 0에 가까워지는 숫자를 표현
    return ( f( x + h ) - f( x ) ) / h
```

```
def f(x):
```



```
return 2*x**2 + 2
```

```
print(numerical_diff( f, 4)) # 0.0
```

진정한 미분은 컴퓨터로 구현할 수가 없다.

컴퓨터로 구현하면 분모가 0이 되어서 계산이 안된다.

$$\text{도함수 공식 : } \lim_{h \rightarrow 0} \frac{f(x+h) - f(x-h)}{(x+h) - (x-h)} = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x-h)}{2h}$$

위의 식을 파이썬으로 구현하면?

```
def numerical_diff( f, x ):
    h = 0.0001
    return ( f( x + h ) - f( x - h ) ) / (2*h)
```

```
def f(x):
    return (2*x**2) + 2
```

```
print(numerical_diff( f, 4)) # 15.999999999998238
```

**문제73. ( 오늘의 마지막 문제 )** 위에서 만든 미분함수를 이용해서 아래의 함수를 미분해서 기울기를 구하는데  $x = 6$  일때의 미분계수를 구하시오!

$$f(x) = 3x^4 + 2x^3 + 6x^2 + 7$$

```
def numerical_diff( f, x ):
    h = 0.0001
    return ( f( x + h ) - f( x - h ) ) / (2*h)
```

```
def f(x):
    return (3*x**4) + (2*x**3) + (6*x**2) + 7
```

# 03/10

2021년 3월 10일 수요일 오전 9:46

## ■ 딥러닝 수업 복습 ( cnn )

1장. numpy 사용법

2장. 퍼셉트론

3장. 3층 신경망 구현 ( 저자가 미리 만들어 온 가중치를 셋팅해서 신경망 구현 )

4장. 2층 신경망 구현 ( 우리가 직접 필기체 데이터를 학습시켜서 신경망 구성 )

1. 오차함수 : 신경망이 뭘 잘못 생각하고 있는지 깨닫게 해주는 함수
2. 미니배치 : 신경망에 데이터를 넣어서 학습시킬 때 조금씩 데이터를 넣어서 학습 시킴
3. 수치미분 : 인공 신경망 구현의 최종 목적은 신경망의 파라미터를 생성  
파라미터 : 가중치, 바이어스

필기체 데이터 분류 ---> 가중치와 바이어스

폐결절과 정상폐 분류 ---> 가중치와 바이어스

- 인공지능의 눈 ( cnn )
  - i. 웹스크롤링 기술
  - ii. 이미지를 신경망에 넣고 학습시킴
- 인공지능의 입과 귀 ( rnn )
  - i. 웹스크롤링 기술
  - ii. 음악, 텍스트( 판결문, 신문기사 )를 신경망에 넣고 학습 시키는 기술

데이터 분석가

딥러닝 개발자( 연구원 )

인공지능 개발자

진정한 접선으로 기울기를 구하는 것은 컴퓨터로 구현할 수 없으니 할선 기울기를 구해야 하는데 이러면 아주 작은 오차가 발생하지만 그 정도 오차는 허용해 줄 수 있다.



w1 = w1-기울기 ----> 기울기가 0이 될때까지 계속 w1을 갱신

w2 = w2-기울기 ----> 기울기가 0이 될때까지 계속 w2를 갱신

가중치를 w1만 갱신하면 안되고 w2도 갱신해줘야하고 w3도 갱신해줘야하기 때문에 편미분이 딥러닝에서 필요하다.

**문제74. 아래의 수학식을 손으로 편미분에서 x1=4 일때와 x2 = 3일때의 기울기를 구하시오!**

$$f = 2x_1^2 + 3x_2^2 + 4$$

$$\frac{\partial f}{\partial x_1} = 4x_1$$

x1에대한 기울기 : 16

x2에대한 기울기 : 18

**문제75. 아래의 수학식을 오차함수로 생성하시오!**

$$f(x_0, x_1) = x_0^2 + x_1^2$$

식 4.6 ( p 125 )

```
import numpy as np
```

```
x = np.array( [ 3.0, 4.0 ] )
```

```
def loss_func(x):  
    return x[0]**2 + x[1]**2
```

```
print( loss_func(x) ) #25.0
```

**문제76. 위의 loss\_func() 함수를 x0=3, x1=4에서 x0에 대해 편미분했을때의 기울기는?**

```
def loss_func(x):  
    return x[0]**2 + x[1]**2  
  
def numerical_diff( f, x ):  
    h = 0.0001  
    return ( f( x + h ) - f( x - h ) ) / (2*h)  
  
def function_tmp1(x0):  
    return x0**2 + 4**2  
  
print( numerical_diff( function_tmp1, 3 ) )
```

결과 :

6.0000000000000378 <--- 6이 나오면 좋은데 뒤에 378이 나온것은 중앙차분오차이다.

**문제77. 위의 loss\_func() 함수를 x0=3, x1=4에서 x1에 대해 편미분했을때의 기울기는?**

```
def loss_func(x):  
    return x[0]**2 + x[1]**2  
  
def numerical_diff( f, x ):  
    h = 0.0001  
    return ( f( x + h ) - f( x - h ) ) / (2*h)  
  
def function_tmp2(x1):  
    return 3**2 + x1**2  
  
print( numerical_diff( function_tmp2, 4 ) ) #7.9999999999999119
```

※ 위의 편미분 방법은 손으로 나머지 하나를 상수화 시켜서 강제로 구현한 코드이므로 파이썬으로 알아서 편미분 되도록 다시 함수로 만들어 본다.

## ■ 편미분하는 numerical\_gradient 함수 만들기 p127

```
import numpy as np
```

```

x = np.array( [3.0, 4.0] )

def numerical_gradient( f, x ):
    h=0.0001
    grad = np.zeros_like(x) # x와 형상이 같은 배열을 생성 [ 0, 0 ]

    for i in range( x.size ): # 0, 1
        tmp_val = x[i] # x[0], 3.0이 tmp_val 에 담긴다.
        x[i] = tmp_val +h # [ 3.0001, 4.0 ]
        fxh1 = f(x) # 3.0001**2 + 4.0**2 = 25.00060001

        x[i] = tmp_val - h # 3.0 - 0.0001 = 2.9999 # [ 2.9999, 4.0 ]
        fxh2 = f(x) # 2.9999**2 + 4.0**2 = 24.99940001

        grad[i] = ( fxh1 - fxh2 ) / ( 2*h ) # ( 25.00060001 - 24.99940001 ) / ( 2*0.0001 )

    x[i] = tmp_val # [ 3.0, 0 ]

```

**문제78. 위의 함수를 방금한 것처럼 다시 디버깅하는데 i가 1일때 디버깅하시오!**

```

import numpy as np

x = np.array( [3.0, 4.0] )

def numerical_gradient( f, x ):
    h=0.0001
    grad = np.zeros_like(x) # x와 형상이 같은 배열을 생성 [ 0, 0 ]

    for i in range( x.size ): # 0, 1
        tmp_val = x[i] # x[1], 4.0이 tmp_val 에 담긴다.
        x[i] = tmp_val +h # [ 3.0, 4.00001 ]
        fxh1 = f(x) # 3.0**2 + 4.0001**2 = 25.00080001

        x[i] = tmp_val - h # 4.0 - 0.0001 = 3.9999 # [ 3.0, 3.9999 ]
        fxh2 = f(x) # 3.0**2 + 3.9999**2 = 24.99920001

        grad[i] = ( fxh1 - fxh2 ) / ( 2*h ) # ( 25.00080001 - 24.99920001 ) / ( 2*0.0001 )
                                                # [ 6.0, 7.9999999999999119]

    x[i] = tmp_val # [ 0, 4.0 ]

```

#####

```

import numpy as np

x = np.array( [3.0, 4.0] )

def numerical_gradient( f, x ):
    h=0.0001
    grad = np.zeros_like(x) # x와 형상이 같은 배열을 생성 [ 0, 0 ]

    for i in range( x.size ): # 0, 1
        tmp_val = x[i] # x[1], 4.0이 tmp_val 에 담긴다.
        x[i] = tmp_val + h # [ 3.0, 4.00001 ]
        fxh1 = f(x) # 3.0**2 + 4.0001**2 = 25.00080001

        x[i] = tmp_val - h # 4.0 - 0.0001 = 3.9999 # [ 3.0, 3.9999 ]
        fxh2 = f(x) # 3.0**2 + 3.9999**2 = 24.99920001

        grad[i] = ( fxh1 - fxh2 ) / ( 2*h ) # ( 25.00080001 - 24.99920001 ) / ( 2*0.0001 )
                                                # [ 6.0, 7.9999999999999119]

        x[i] = tmp_val # [ 0, 4.0 ]
    return grad

print(numerical_gradient( loss_func, np.array([3.0, 4.0]) )) #[6. 8.]

```

## ■ 경사하강법 ( p 129 )

방금 만든 numerical\_gradient 함수는 산에서 내려오기 위해서 내가 서있는 곳에서 어느쪽으로 가야 산 아래로 내려갈 수 있는지 내가 서있는 곳의 기울기를 구하는 함수이다.

가중치 = 가중치 - 기울기

위의 식을 loop문으로 계속 반복해서 수행해서 기울기가 0이 되면 가중치가 변경이 안되므로 그 시점까지 수행을 해서 최적의 가중치를 알아낼 것이다.

가중치 = 가중치 - 학습률( LR ) \* 기울기

학습률은 알아서 정해줘야하는데 너무 크거나 작으면 global minima를 찾아갈 수 없다.

학습률이 너무 크면 학습은 빠르지만 global minimum을 지나칠 수 있다.

학습률이 너무 작으면 global minimum을 지나칠 염려는 없지만 학습이 너무 느려 수렴을 못한다.

**문제79. 방금 만든 numerical\_gradient 함수를 이용해서 경사 하강을 하는**

## gradient\_decent 함수를 생성하시오! p 131

```
def gradient_decent( f, init_x, lr=0.01, step_num = 100 ) :  
    x = init_x  
    for i in range( step_num ):  
        grad = numerical_gradient( f, x )  
        x -= lr*grad  
    return x
```

**문제80. 책 132처럼 함수  $f(x_0, x_1) = x_0^2 + x_1^2$  함수를 오차함수로 두고 처음 지점을  $[-3.0, 4.0]$  하고 계속 경사하강을 해서 최소지점인  $[0, 0]$  지점으로 경사하강되게 하시오!**

```
def loss_func(x):  
    return x[0]**2 + x[1]**2  
  
init_x = np.array( [-3.0, 4.0] )  
  
a = gradient_descent( loss_func, init_x, lr=0.1, step_num = 100)  
  
print(a.round()) # [-0. 0.]
```

**문제81. ( 점심시간문제 )** 러닝레이트를 0.1로 하지말고 너무 작은 값을 주어서 최소지점에 도달하지 못하는지 확인하시오!

```
def loss_func(x):  
    return x[0]**2 + x[1]**2  
  
init_x = np.array( [-3.0, 4.0] )  
  
a = gradient_descent( loss_func, init_x, lr=1e-50, step_num = 100)  
  
print(a.round())
```

3장에서 만들었던 신경망은 오로지 안루쿰 교수님이 준비해주신 필기체 데이터만 분류할 수 있었다.

목표 : 00 신경망 생성

설현사진과 아이린 사진을 분류하는 신경망을 만들고 싶다면 ?

어제 만들었던 3층신경망에 설현사진을 넣고 숫자를 뭘로 예측하는지 ?



## ■ 설현 사진 resize 하고 흑백처리 (신경망에 입력되기 전 데이터 구성)

# 1. d:WWdata10 폴더 아래에 a.jpg를 가져다 둔다.

# 2. 아나콘다 프롬프트 창을 열고 cv2 모듈을 설치한다.

```
pip install opencv-python
```

# 3. D:WWdata10 폴더 밑에 있는 파일을 불러온다.

```
import numpy as np
```

```
import os
```

```
import cv2
```

```
path = "D:WWdata10"
```

```
file_list = os.listdir(path)
```

```
file_list
```

# 4. 설현사진을 128x128로 resize 한다.

```
for k in file_list: # 리스트 안에 있는 파일들을 하나씩 빼내는 코드
```

```
    img = cv2.imread(path + 'WW' + k) # 설현 사진을 숫자행렬로 변경한다.
```

```
    print( img )
```

```
for k in file_list: # 리스트 안에 있는 파일들을 하나씩 빼내는 코드
```

```
    img = cv2.imread(path + 'WW' + k) # 설현 사진을 숫자행렬로 변경한다.
```

```
    width, height = img.shape[:2] # 설현사진 숫자 행렬에서 가로, 세로를 가져온다.
```

```
    resize_img = cv2.resize(img, (128 , 128), interpolation=cv2.INTER_CUBIC)
```

```
    cv2.imwrite('d:WWdata11WWresizeWW' + k, resize_img) # resize한 이미지를 저장한다.
```

```
plt.imshow(resize_img) # resize 된 사진을 시각화
```

```
plt.show()
```

# 5. resize된 설현 사진을 흑백으로 변경한다.

```
j= 'd:WWdata11WWresizeWWa.jpg'
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import matplotlib.image as mpimg
```

```
def rgb2gray(rgb):
    return np.dot(rgb[...,:3], [0.299, 0.587, 0.114])
```

```
img = mpimg.imread(j)
gray = rgb2gray(img)
plt.imshow(gray, cmap = plt.get_cmap('gray'))
plt.show()
```

#####

### # 1. 설현사진을 숫자로 변경합니다.

```
import cv2
import os
import numpy as np

path = "D:\WWdata10"
file_list = os.listdir(path)
for k in file_list:
    img = cv2.imread(path + '\WW' + k)
    print(img )
```

### # 2. 설현사진을 resize 합니다.

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

for k in file_list:
    img = cv2.imread(path + '\WW' + k)
    width, height = img.shape[:2]
    resize_img = cv2.resize(img, (28 , 28), interpolation=cv2.INTER_CUBIC)
    cv2.imwrite('d:\WWdata11\WWresize\WW' + k, resize_img)

plt.imshow(resize_img)
plt.show()
```

### 3. 설현 사진을 흑백으로 변경합니다.

```
j= 'd:\WWdata11\WWresize\WWa.jpg'
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

def rgb2gray(rgb):
    return np.dot(rgb[...,:3], [0.299, 0.587, 0.114])

img = mpimg.imread(j)
gray = rgb2gray(img)
plt.imshow(gray, cmap = plt.get_cmap('gray'))
```

```
plt.show()
```

**# 설현사진을 1차원으로 변경해주기**

```
gray.shape
x=gray.flatten()
x.shape
```

**# 4. 설현 사진을 mnist 신경망에 넣습니다.**

```
import numpy as np
import pickle
from common import sigmoid, softmax
from dataset.mnist import load_mnist
```

**# 1. 파이썬 기초 ---> 2. 함수를 생성 ----> 3. 함수로 클래스를 생성**

```
class Three_nn():
    import numpy as np
    import pickle
    from common import sigmoid, softmax
    from dataset.mnist import load_mnist

    def init_network(self):
        import pickle
        with open("C:\\Users\\stuu03\\deep-learning-from-scratch-master\\ch03\\sample_weight.pkl", "rb") as f:
            network = pickle.load(f)
        return network
```

**# 1. 데이터를 불러옵니다. ( 안르쿰 교수님이 만든 필기체 데이터)**

```
def get_data(self):
    (x_train, t_train), (x_test, t_test) = load_mnist(flatten=True, normalize=True,
one_hot_label=False)
    return x_train, t_train
```

**# 2. 가중치와 바이어스 값을 불러와서 3층 신경망에 흘려보내는 함수**

```
def predict(self, network, x):
    #network = init_network()
    w1, w2, w3 = network['W1'], network['W2'], network['W3']
    b1, b2, b3 = network['b1'], network['b2'], network['b3']
```

**# 3. 신경망을 구성합니다.**

**# 0층**

**# x = x\_train[0:100] # 일단 10개의 필기체 데이터를 구성합니다.**

**# 1층**

```
y = np.dot(x,w1) + b1
y_hat = sigmoid(y)
```

**# 2층**

```

z = np.dot(y_hat, w2) + b2
z_hat = sigmoid(z)
# 3층
k = np.dot(z_hat, w3) + b3
k_hat = softmax(k)
return k_hat

```

n1 = Three\_nn() # 객체화 시킨다. 설계도 가지고 제품을 만든다.

```

#x, t = n1.get_data() # 테스트 데이터와 테스트 데이터의 정답을 불러오는 코드
network = n1.init_network() # 저자가 만들어온 가중치와 바이어스를 불러오는 코드

```

```

result = n1.predict( network, x )
print (np.argmax(result) ) #3

```

**문제82. 이번에는 다른 설현사진을 신경망에 넣고 어떤 숫자가 나오는지 확인 하시오!**

위랑 똑같이 하되 사진만 변경할 것

## ■ 편미분을 하는 **gradient\_decent** 함수를 이용해서 2층 신경망을 구현

저자가 만들어온 소스 코드내에 common이라는 폴더가 있다.

이 common 폴더를 쥬피터 워킹 디렉토리에 갖다 놓기

기존에 있었던 common.py는 common7.py로 이름을 변경하시오!

common 폴더( 패키지 ) 안에는 functions.py도 있고 gradient.py도 있다.

functions.py --> 신경망에 필요한 함수들

gradient.py --> 편미분하는 함수 numerical\_gradient 함수

### # 2층 신경망 만들기

```

import sys, os
sys.path.append( os.pardir ) # 부모디렉토리의 파일들을 가져올 수 있도록 설정
import numpy as np
from common.functions import softmax, cross_entropy_error

```

```
from common.gradient import numerical_gradient
```

```
z = np.array( [0.1, 0.9] )  
print( softmax(z) )
```

### # 예제1. 가중치 행렬을 2x3으로 랜덤으로 숫자를 생성해서 생성하기

```
import numpy as np  
print( np.random.randn(2,3) ) # [[ 0.22125821 -0.45423139 -0.13062317]  
                                # [-2.7799301  0.34189776  2.67812111]]
```

### # 예제2. 아래의 입력 데이터를 1x2 행렬로 만들고 위에서 만든 가중치 행렬과 내적하시오!

```
x = np.array( [0.6, 0.9] ) # 설현사진  
W = np.random.randn(2,3)  
print( np.dot( x, W ) )
```

### # 예제3. 설현 사진을 가중치 행렬과 내적해서 나온 결과 행렬을 softmax 함수에 넣어서 확률을 출력하시오!

```
x = np.array( [0.6, 0.9] ) # 설현사진  
W = np.random.randn(2,3)  
z = ( np.dot( x, W ) )  
y = softmax(z)  
print(y) # [0.14845871 0.27518052 0.57636077]
```

### # 예제4. 위에 출력된 확률 벡터를 정답과 함께 오차함수에 넣어서 오차를 구해본다.

```
x = np.array( [0.6, 0.9] ) # 설현사진  
np.random.seed(1)  
W = np.random.randn(2,3)  
t = np.array( [0,1,0] ) # 두번째가 1이면 설현 사진 ( 정답 )  
z = ( np.dot( x, W ) )  
y = softmax(z)  
print(y) # [0.14845871 0.27518052 0.57636077]  
loss = cross_entropy_error( y, t )  
print( loss ) #0.5476574632667771
```

## # 예제5. 비용함수를 생성하시오!

```
f = lambda w : net.loss( x, t ) #비용함수 생성
dW = numerical_gradient( f, net.W )
print( dW ) #기울기 출력
```

## # 예제6. 2층 신경망 전체코드 구현

```
import sys, os
sys.path.append( os.pardir ) # 부모디렉토리의 파일들을 가져올 수 있도록 설정
import numpy as np
from common.functions import softmax, cross_entropy_error
from common.gradient import numerical_gradient

class simpleNet:
    def __init__(self): # 설계도( 클래스 )로 제품( 객체 )을 만들 때 바로 실행되는 함수
        self.W = np.random.randn( 2,3 ) # 랜덤으로 가중치 행렬을 생성

    def predict( self, x ):
        return np.dot( x, self.W )

    def loss( self, x, t ):
        z = self.predict(x) #설현 사진을 넣어서 z 값을 출력
        y = softmax(z) # z값을 받아서 확률벡터 출력
        loss = cross_entropy_error( y, t ) # 확률벡터와 정답을 넣어서 오차 출력
        return loss

x = np.array( [0.6, 0.9] )
t = np.array( [0, 0, 1] ) # [아이린, 설현, 아이유]

net = simpleNet() #클래스( 2층 신경망 설계도)로 객체( 제품 )를 생성

f = lambda w : net.loss( x, t ) # 비용함수 생성 / 이름없는 한줄짜리 함수

dW = numerical_gradient( f, net.W )
print(dW)
```

결과 :

```
[[ 0.28292283  0.29350464 -0.57642747]
 [ 0.42438424  0.44025696 -0.8646412 ]]
```

설명 : 가중치가 2x3행렬이니까 기울기도 2x3행렬로 나와야 가중치에서 기울기를 뺄 수 있다.

가중치 = 가중치 - 기울기

## ■ 딥러닝 책 내용

### 1장. numpy 사용법 ?

신경망에서 예측하는 모든 수학식이 행렬 계산이어서 numpy가 행렬 연산을 고속으로 처리하는 모듈

### 2장. 퍼셉트론 ?

인공신경망의 원리 ---> 가중치를 갱신하는 작업

### 3장. 3층 신경망 구현 : 필기체 데이터를 분류하는 인공신경망을 3층으로 생성

( 저자가 필기체 6만장을 학습 다 시킨 가중치를 만들어와서 제공해줘서 우리는 그냥 피클 파일 불러와서 3층 신경망 구성 )

설현사진 ----> 3층 신경망( 필기체를 공부한 신경망 )에 넣기 ----> 결과 : 8

### 4장. 2층 신경망 구현 ( 학습을 시켜야 함 )

1. 오차함수
2. 미니배치
3. 수치미분 ----> 오차함수를 미분해서 기울기를 구하려고

입력값 -----> 2층 신경망 -----> 기울기

↓

1. `__init__` 함수 : 가중치 행렬을 생성
2. `predict` 함수 : 입력값 받아서 가중치 행렬과 내적인 결과를 출력
3. `loss` 함수 : `predict` -> `softmax` -> `cross_entropy_error` -> 오차

```
x = np.array( [0.6, 0.9] ) # 설현 사진, 1x2
self.W = np.random.randn( 2,3 ), 2x3
np.dot( x, self.W ) ---> 1x3
```

```
net1 = simpleNet()
가중치 = 가중치( 2x3 ) - 기울기( 2x3 )
net = SimpleNet()
```

```
f = lambda w: net.loss( x, t ) # 오차함수
numerical_gradient( f, net.W )
```

**문제83. 아래의 입력 데이터와 target( 정답 )을 simplenet에 입력하고 오차를 출력하시오!**

```
x = np.array( [ 0.8, 0.2 ] )
t = np.array( [ 0, 0, 1 ] )
```

답 :

```
x = np.array( [ 0.8, 0.2 ] )
t = np.array( [ 0, 0, 1 ] )
```

```
net = simpleNet() # 클래스(설계도 )로 객체( 제품 )를 생성합니다.
```

```
print( net.loss( x, t ) ) #0.4948858718953349
```

**문제84. simpleNet()클래스에 있는 가중치 행렬 W를 출력하시오!**

```
net = simpleNet()
print( net.W )
```

결과 :

```
[[ 0.04221375  0.58281521 -1.10061918]
 [ 1.14472371  0.90159072  0.50249434]]
```

**문제85. simpleNet() 클래스에 있는 predict 함수에 아래의 입력 데이터를 넣고 결과를 출력하시오!**

```
x = np.array( [ 0.9, 0.6 ] )
```

```
net = simpleNet()
x = np.array( [ 0.9, 0.6 ] )
print( net.predict(x) ) #[ -1.12961806 -0.75982585 -0.62605419 ]
```

#####

```
net = simpleNet()
x = np.array( [ 0.9, 0.6 ] )
result = net.predict(x)
print( result )
```



# ■ 필기체 데이터를 학습 시키는 2층 신경망 전체 full 코드( p 137 )

소스 코드는 ch04/two\_layer\_net.py

## # 1. 필요한 신경망 구현 함수들 가져오는 코드

```
# coding: utf-8
import sys, os
sys.path.append(os.pardir) # 부모 디렉터리의 파일을 가져올 수 있도록 설정
from common.functions import *
from common.gradient import numerical_gradient
```

## # 2. 2층 신경망 구현하는 클래스 생성 코드

```
class TwoLayerNet:
```

## # 3. 가중치 행렬 생성하는 코드

### # 2층 신경망에는 가중치 행렬이 몇개가 필요할까?

```
def __init__(self, input_size, hidden_size, output_size, weight_init_std=0.01):
    # 가중치 초기화
    self.params = {}
    self.params['W1'] = weight_init_std * np.random.randn(input_size, hidden_size)
    self.params['b1'] = np.zeros(hidden_size)
    self.params['W2'] = weight_init_std * np.random.randn(hidden_size, output_size)
    self.params['b2'] = np.zeros(output_size)
```

## # 4. 입력값을 받아서 가중치 행렬과 내적하고 바이어스를 더해서 나온 행렬을 소프트 맥스 함수에 넣어서 확률벡터를 출력하는 함수가 predict 함수 이다.

```
def predict(self, x):
    W1, W2 = self.params['W1'], self.params['W2']
    b1, b2 = self.params['b1'], self.params['b2']

    a1 = np.dot(x, W1) + b1 # 1층 구현
    z1 = sigmoid(a1)         # 시그모이드 함수 통과
    a2 = np.dot(z1, W2) + b2 # 2층 구현
    y = softmax(a2)          # 소프트맥스 함수 통과

    return y # [0.1, 0.1, 0.1, 0.3, 0.1, 0.1, 0.05, 0.05, 0.05, 0.05 ]
```

```

# x : 입력 데이터, t : 정답 레이블
def loss(self, x, t):
    y = self.predict(x)

    return cross_entropy_error(y, t)

def accuracy(self, x, t):
    y = self.predict(x)
    y = np.argmax(y, axis=1)
    t = np.argmax(t, axis=1)

    accuracy = np.sum(y == t) / float(x.shape[0])
    return accuracy

# x : 입력 데이터, t : 정답 레이블
def numerical_gradient(self, x, t):
    loss_W = lambda W: self.loss(x, t)

    grads = {}
    grads['W1'] = numerical_gradient(loss_W, self.params['W1'])
    grads['b1'] = numerical_gradient(loss_W, self.params['b1'])
    grads['W2'] = numerical_gradient(loss_W, self.params['W2'])
    grads['b2'] = numerical_gradient(loss_W, self.params['b2'])

    return grads

def gradient(self, x, t):
    W1, W2 = self.params['W1'], self.params['W2']
    b1, b2 = self.params['b1'], self.params['b2']
    grads = {}

    batch_num = x.shape[0]

    # forward
    a1 = np.dot(x, W1) + b1
    z1 = sigmoid(a1)
    a2 = np.dot(z1, W2) + b2
    y = softmax(a2)

    # backward
    dy = (y - t) / batch_num
    grads['W2'] = np.dot(z1.T, dy)
    grads['b2'] = np.sum(dy, axis=0)

    da1 = np.dot(dy, W2.T)
    dz1 = sigmoid_grad(a1) * da1
    grads['W1'] = np.dot(x.T, dz1)
    grads['b1'] = np.sum(dz1, axis=0)

    return grads

```

**문제86.** 오늘 만든 simpleNet 클래스를 객체화 시켜서 실행하는데 이 신경망에 설현사진을 입력할 수 있도록 가중치 행렬의 shape를 수정하고 설현사진을 입력해서 확률벡터를 출력하시오! ( 설현 사진을 28x28로 리사이즈 해서 넣으시오

( 784, 50 ) 행렬이어야 함

# 4. 입력값을 받아서 가중치 행렬과 내적하고 바이어스를 더해서 나온 행렬을 소프트 맥스 함수 넣어서 확률벡터를 출력하는 함수가 predict 함수 입니다.

```
def predict(self, x):
    W1, W2 = self.params['W1'], self.params['W2']
    b1, b2 = self.params['b1'], self.params['b2']

    a1 = np.dot(x, W1) + b1 # 1층 구현
    z1 = sigmoid(a1) # 시그모이드함수 통과
    a2 = np.dot(z1, W2) + b2 # 2층 구현
    y = softmax(a2) # 소프트맥스 함수 통과

    return y # [0.1, 0.1, 0.1, 0.3, 0.1, 0.1, 0.05, 0.05, 0.05, 0.05]
```

# x : 입력 데이터, t : 정답 레이블

# 1. 설현사진을 숫자로 변경합니다.

```
import cv2
import os
import numpy as np

path = "D:WWdata10"
file_list = os.listdir(path)
for k in file_list:
    img = cv2.imread(path + 'WW' + k)
    print(img )

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

for k in file_list:
    img = cv2.imread(path + 'WW' + k)
    width, height = img.shape[:2]
    resize_img = cv2.resize(img, (28 , 28), interpolation=cv2.INTER_CUBIC)
    cv2.imwrite('d:WWdata11WWresizeWW' + k, resize_img)

plt.imshow(resize_img)
plt.show()

# 3. 설현 사진을 흑백으로 변경합니다.
```

```
j= 'd:\wwdata1\wwresize\wwa.jpg'
```

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

```
def rgb2gray(rgb):
    return np.dot(rgb[...,:3], [0.299, 0.587, 0.114])
```

```
img = mpimg.imread(j)
gray = rgb2gray(img)
plt.imshow(gray, cmap = plt.get_cmap('gray'))
plt.show()
```

**# 설현사진을 1차원으로 변경해주기**

```
gray.shape
x=gray.flatten()
x.shape
```

```
import numpy as np
import pickle
from common7 import sigmoid, softmax
from dataset.mnist import load_mnist
```

**# 1. 파이썬 기초 ---> 2. 함수를 생성 ----> 3. 함수로 클래스를 생성**

```
class Three_nn():
    import numpy as np
    import pickle
    from common7 import sigmoid, softmax
    from dataset.mnist import load_mnist

    def init_network(self):
        import pickle
        with open("C:\wwUsers\wwstu03\wwdeep-learning-from-scratch-master\wwch03\wwsample_weight.pkl", "rb") as f:
            network = pickle.load(f)
        return network
```

**# 1. 데이터를 불러옵니다. ( 안르쿰 교수님이 만든 필기체 데이터)**

```
def get_data(self):
    (x_train, t_train), (x_test, t_test) = load_mnist(flatten=True, normalize=True,
one_hot_label=False)
    return x_train, t_train
```

**# 2. 가중치와 바이어스 값을 불러와서 3층 신경망에 흘려보내는 함수**

```
def predict(self, network, x):
    #network = init_network()
    w1, w2, w3 = network['W1'], network['W2'], network['W3']
    b1, b2, b3 = network['b1'], network['b2'], network['b3']
```

**# 3. 신경망을 구성합니다.**

```

# 0층
# x = x_train[0:100] # 일단 10개의 필기체 데이터를 구성합니다.
# 1층
y = np.dot(x,w1) + b1
y_hat = sigmoid(y)
# 2층
z = np.dot(y_hat, w2) + b2
z_hat = sigmoid(z)
# 3층
k = np.dot(z_hat, w3) + b3
k_hat = softmax(k)
return k_hat

```

n1 = Three\_nn() # 객체화 시킨다. 설계도 가지고 제품을 만든다.

```

#x, t = n1.get_data() # 테스트 데이터와 테스트 데이터의 정답을 불러오는 코드
network = n1.init_network() # 저자가 만들어온 가중치와 바이어스를 불러오는 코드

```

```

result = n1.predict( network, x )
print (np.argmax(result) ) #3

```

```

import sys, os
sys.path.append(os.pardir) # 부모디렉토리의 파일들을 가져올 수 있도록 설정
import numpy as np
from common.functions import softmax, cross_entropy_error
from common.gradient import numerical_gradient

```

```

class simpleNet:
    def __init__(self): # 설계도(클래스)로 제품(객체)을 만들때 바로 실행되는 함수
        self.W = np.random.randn(784,3) # 랜덤으로 가중치 행렬을 생성

    def predict(self, x): # 설현 사진 입력해서 예측하는 함수
        return np.dot( x, self.W)

    def loss(self, x, t):
        z = self.predict(x) # 설현 사진을 넣어서 z 값을 출력
        y = softmax(z) # z 값 받아서 확률벡터를 출력
        loss = cross_entropy_error(y, t) # 확률벡터와 정답을 넣어서 오차를 출력
        return loss

```

```

net = simpleNet()
result = net.predict(x)
print ( result ) # [-2034.18660908  9915.07165935 -442.68328935]

```

문제86. ( 오늘의 마지막 문제 ) 설현 사진을 simpleNet 신경망에 넣으시오!

# 1. 설현사진을 숫자로 변경합니다.

```
import cv2
```

```
import os
```

```
import numpy as np
```

```
path = "D:WWdata10"
```

```
file_list = os.listdir(path)
```

```
for k in file_list:
```

```
    img = cv2.imread(path + 'WW' + k)
```

```
print(img )
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import matplotlib.image as mpimg
```

```
for k in file_list:
```

```
    img = cv2.imread(path + 'WW' + k)
```

```
    width, height = img.shape[:2]
```

```
    resize_img = cv2.resize(img, (28 , 28), interpolation=cv2.INTER_CUBIC)
```

```
    cv2.imwrite('d:WWdata11WWresizeWW' + k, resize_img)
```

```
plt.imshow(resize_img)
```

```
plt.show()
```

# 3. 설현 사진을 흑백으로 변경합니다.

```
j= 'd:WWdata11WWresizeWWa.jpg'
```

```
import numpy as np
```

```
import matplotlib.pyplot as plt
```

```
import matplotlib.image as mpimg
```

```
def rgb2gray(rgb):
```

```
    return np.dot(rgb[...,:3], [0.299, 0.587, 0.114])
```

```
img = mpimg.imread(j)
```

```
gray = rgb2gray(img)
```

```
plt.imshow(gray, cmap = plt.get_cmap('gray'))
```

```
plt.show()
```

# 설현사진을 1차원으로 변경해주기

```
gray.shape
```

```
x=gray.flatten()
```

```
x.shape
```

```
import numpy as np
```

```
import pickle
```

```
from common7 import sigmoid, softmax
```

```
from dataset.mnist import load_mnist
```

```
import numpy as np
```

```
import pickle
```

```
from common7 import sigmoid, softmax
from dataset.mnist import load_mnist
```

# 1. 파이썬 기초 ---> 2. 함수를 생성 ----> 3. 함수로 클래스를 생성

```
class Three_nn():
    import numpy as np
    import pickle
    from common7 import sigmoid, softmax
    from dataset.mnist import load_mnist

    def init_network(self):
        import pickle
        with open("C:\\Users\\Wst03\\deep-learning-from-scratch-master\\ch03\\sample_weight.pkl", "rb") as f:
            network = pickle.load(f)
        return network
```

# 1. 데이터를 불러옵니다. ( 안르쿤 교수님이 만든 필기체 데이터)

```
def get_data(self):
    (x_train, t_train), (x_test, t_test) = load_mnist(flatten=True, normalize=True,
one_hot_label=False)
    return x_train, t_train
```

# 2. 가중치와 바이어스 값을 불러와서 3층 신경망에 흘려보내는 함수

```
def predict(self, network, x):
    #network = init_network()
    w1, w2, w3 = network['W1'], network['W2'], network['W3']
    b1, b2, b3 = network['b1'], network['b2'], network['b3']
```

# 3. 신경망을 구성합니다.

# 0층

# x = x\_train[0:100] # 일단 10개의 필기체 데이터를 구성합니다.

# 1층

y = np.dot(x, w1) + b1

y\_hat = sigmoid(y)

# 2층

z = np.dot(y\_hat, w2) + b2

z\_hat = sigmoid(z)

# 3층

k = np.dot(z\_hat, w3) + b3

k\_hat = softmax(k)

return k\_hat

n1 = Three\_nn() # 객체화 시킨다. 설계도 가지고 제품을 만든다.

#x, t = n1.get\_data() # 테스트 데이터와 테스트 데이터의 정답을 불러오는 코드

network = n1.init\_network() # 저자가 만들어온 가중치와 바이어스를 불러오는 코드

```
result = n1.predict( network, x )  
print (np.argmax(result) ) #3
```

```
import sys, os  
sys.path.append(os.pardir) # 부모디렉토리의 파일들을 가져올 수 있도록 설정  
import numpy as np  
from common.functions import softmax, cross_entropy_error  
from common.gradient import numerical_gradient  
  
class simpleNet:  
    def __init__(self): # 설계도(클래스)로 제품(객체)을 만들때 바로 실행되는 함수  
        self.W = np.random.randn(784,3) # 랜덤으로 가중치 행렬을 생성  
  
    def predict(self, x): # 설현 사진 입력해서 예측하는 함수  
        return np.dot( x, self.W)  
  
    def loss(self, x, t):  
        z = self.predict(x) # 설현 사진을 넣어서 z 값을 출력  
        y = softmax(z) # z 값 받아서 확률벡터를 출력  
        loss = cross_entropy_error(y, t) # 확률벡터와 정답을 넣어서 오차를 출력  
        return loss  
  
net = simpleNet()  
result = net.predict(x)  
print ( result ) # [-2034.18660908  9915.07165935 -442.68328935]
```



03/11

2021년 3월 11일 목요일 오전 9:42

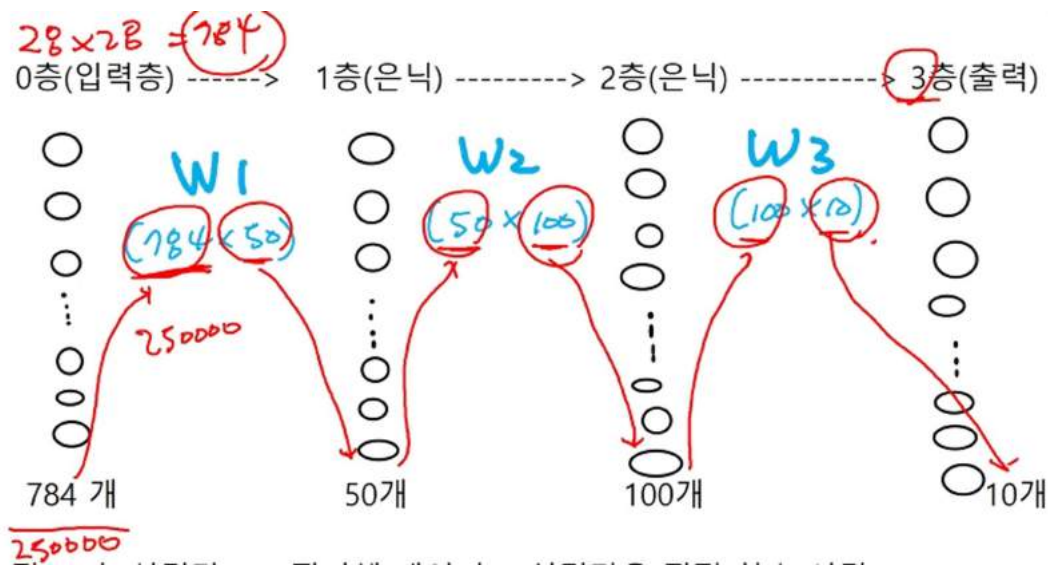
## ■ 딥러닝 수업 복습 :

딥러닝이 무엇인가요? 답변을 세련되고 정중하게 대답할 수 있도록 생각하기

1장. numpy를 왜 사용해야 하는지?

2장. 퍼셉트론

3장. 3층 신경망 --> 저자가 만들어 온 가중치를 셋팅해서 필기체를 분류하는 신경망



4장. 2층 신경망 --> 필기체 데이터로 신경망을 직접 학습 시킴 ( p 143 )

--> 설현과 수지사진을 분류하는 신경망

5장. 오차 역전파

6장. 언더피팅과 오버피팅을 해결하는 방법들

7장. CNN

8장. 최신 기술들 소개 --> tensorflow, pytorch로 신경망 구현

## ■ ( 필기체 데이터를 인식하게끔 학습시키는 ) 2층 신경망 구현하기 p137-144

1. 4장 전체코드를 먼저 수행합니다.

**문제87. TwoLayerNet 클래스를 객체화 시켜서 numerical\_gradient 함수를 실행해 기울기를 출력하시오! ( 입력데이터를 100개만 입력하시오! )**

**문제88. ( 점심시간 문제 ) 10에폭이아 니라 20에폭둘게 코드를 수정하여 돌리고 시각화 된 그래프를 첨부하세요!**

```
# coding: utf-8
import sys, os
sys.path.append(os.pardir) # 부모 디렉터리의 파일을 가져올 수 있도록 설정
from common.functions import *
from common.gradient import numerical_gradient

from dataset.mnist import load_mnist
class TwoLayerNet:
## 1. 가중치 행렬 W1, W2, b1, b2를 구성한다.
    def __init__(self, input_size, hidden_size, output_size, weight_init_std=0.01):
        # 가중치 초기화
        self.params = {}
        self.params['W1'] = weight_init_std * np.random.randn(input_size, hidden_size)
        self.params['b1'] = np.zeros(hidden_size)
        self.params['W2'] = weight_init_std * np.random.randn(hidden_size, output_size)
        self.params['b2'] = np.zeros(output_size)
        print('신경망이 생성되었습니다.')

#hidden_size만 다른 숫자로 바꿀 수 있다. input이나 output을 바꾸면 에러가 난다.
#network = TwoLayerNet(input_size=784, hidden_size=50, output_size=10)
#network.params.keys() # dict_keys(['W1', 'b1', 'W2', 'b2'])
# 가중치 행렬을 가지고 있는 params 딕셔너리의 키값들 확인
#network.params['W1'].shape # (784, 50)
#network.params['W2'].shape # (50, 10)
#network.params['b1'].shape # (50,)
#network.params['b2'].shape # (10,)

# 2. 입력데이터( 필기체 )를 넣고 1층과 2층을 거쳐서 확률벡터를 출력하는 함수

    def predict(self, x):
        W1, W2 = self.params['W1'], self.params['W2'] # 가중치 불러오는 코드
        b1, b2 = self.params['b1'], self.params['b2'] # 바이어스 불러오는 코드
```

```

a1 = np.dot(x, W1) + b1 # 1층 구성
z1 = sigmoid(a1)        # 1층 시그모이드 함수
a2 = np.dot(z1, W2) + b2 # 2층 구성
y = softmax(a2)         # 2층이 출력층이라 소프트맥스 함수

return y

#(x_train, t_train), (x_test, t_test) = load_mnist(normalize=True, one_hot_label=True)

#network = TwoLayerNet(input_size=784, hidden_size=50, output_size=10)
#network.predict(x_train).shape #(60000, 10)

# 3. 오차(에러)를 출력하는 함수
# x : 입력 데이터, t : 정답 레이블
def loss(self, x, t):
    y = self.predict(x)

    return cross_entropy_error(y, t)

# network = TwoLayerNet(input_size=784, hidden_size=50, output_size=10)
# (x_train, t_train), (x_test, t_test) = load_mnist(normalize=True, one_hot_label=True)
# network.loss(x_train[100], t_train[100]) #2.287963096167366

# 4. 정확도를 출력하는 함수
def accuracy(self, x, t):
    y = self.predict(x)
    y = np.argmax(y, axis=1)
    t = np.argmax(t, axis=1)

    accuracy = np.sum(y == t) / float(x.shape[0])
    return accuracy

#network = TwoLayerNet(input_size=784, hidden_size=50, output_size=10)
#(x_train, t_train), (x_test, t_test) = load_mnist(normalize=True, one_hot_label=True)
#network.accuracy(x_train[:100], t_train[:100]) #0.1

#5. 편미분해서 기울기를 출력하는 함수( 4개의 기울기를 출력, w1, b1, w2, b2)

# x : 입력 데이터, t : 정답 레이블
def numerical_gradient(self, x, t):
    loss_W = lambda W: self.loss(x, t)

    grads = {}
    grads['W1'] = numerical_gradient(loss_W, self.params['W1'])
    grads['b1'] = numerical_gradient(loss_W, self.params['b1'])
    grads['W2'] = numerical_gradient(loss_W, self.params['W2'])
    grads['b2'] = numerical_gradient(loss_W, self.params['b2'])

```

```
return grads
```

```
#network = TwoLayerNet(input_size=784, hidden_size=50, output_size=10)
#(x_train, t_train), (x_test, t_test) = load_mnist(normalize=True, one_hot_label=True)
#network.numerical_gradient(x_train[:100,], t_train[:100,]) #0.1
```

#6. 위의 수치미분은 너무 느려서 못 쓰고 5장에서 배울 오차역전파 써서 가중치를 해줘야 한다.  
# 아래의 gradient는 5장에서 배울 오차 역전파 함수이다.  
# 이게 학습이 훨씬 빠르다.

```
def gradient(self, x, t):
    W1, W2 = self.params['W1'], self.params['W2']
    b1, b2 = self.params['b1'], self.params['b2']
    grads = {}
```

```
    batch_num = x.shape[0]
```

```
    # forward
    a1 = np.dot(x, W1) + b1
    z1 = sigmoid(a1)
    a2 = np.dot(z1, W2) + b2
    y = softmax(a2)
```

```
    # backward
    dy = (y - t) / batch_num
    grads['W2'] = np.dot(z1.T, dy)
    grads['b2'] = np.sum(dy, axis=0)
```

```
    da1 = np.dot(dy, W2.T)
    dz1 = sigmoid_grad(a1) * da1
    grads['W1'] = np.dot(x.T, dz1)
    grads['b1'] = np.sum(dz1, axis=0)
```

```
    return grads
```

```
network = TwoLayerNet(input_size=784, hidden_size=50, output_size=10)
(x_train, t_train), (x_test, t_test) = load_mnist(normalize=True, one_hot_label=True)
network.gradient(x_train[:100,], t_train[:100,]) #0.1
```

### 지금부터의 코드는 위에서 만든 클래스를 객체화 시키고 필기체 데이터를 불러와서 객체화  
시킨 신경망에 100개씩  
### 입력해서 학습 시키는 코드

```
# coding: utf-8
import sys, os
```

```

sys.path.append(os.pardir) # 부모 디렉터리의 파일을 가져올 수 있도록 설정
import numpy as np
import matplotlib.pyplot as plt #정확도를 시각화 하기 위해 필요
from dataset.mnist import load_mnist #필기체 데이터를 불러오는 코드
#from two_layer_net import TwoLayerNet

# 데이터 읽기
(x_train, t_train), (x_test, t_test) = load_mnist(normalize=True, one_hot_label=True)

network = TwoLayerNet(input_size=784, hidden_size=50, output_size=10)

# 하이퍼파라미터
iters_num = 12000 # 반복 횟수를 적절히 설정한다.10에폭 돌게 설정 ( 6000x10 = 60000)
train_size = x_train.shape[0]
batch_size = 100 # 미니배치 크기
learning_rate = 0.1 #학습률

train_loss_list = [] # 오차를 담을 리스트 ( 시각화를 위해 데이터 저장)
train_acc_list = [] # 훈련 데이터의 정확도를 담을 리스트( 시각화를 위해 데이터 저장)
test_acc_list = [] # 테스트 데이터의 정확도를 담을 리스트( 시각화를 위해 데이터 저장)

# 1에폭당 반복 수
iter_per_epoch = max(train_size / batch_size, 1) # (60000 / 100 = 600 ) 또는 1중에 큰 값 선택
# 1에폭당 정확도를 시각화 하기 위해 필요한 코드

for i in range(iters_num): # 6000
    # 미니배치 획득 # 60000, 100개 / 0~ 60000 미만까지의 숫자중
    batch_mask = np.random.choice(train_size, batch_size) # 100개의 숫자를 랜덤 추출
    x_batch = x_train[batch_mask] # 훈련데이터 100개
    t_batch = t_train[batch_mask] # 훈련데이터 정답 100개

    # 기울기 계산
    #grad = network.numerical_gradient(x_batch, t_batch)
    grad = network.gradient(x_batch, t_batch) # 기울기 100개를 구한다.

    # 매개변수 갱신
    for key in ('W1', 'b1', 'W2', 'b2'): #가중치와 바이어스를 갱신
        network.params[key] -= learning_rate * grad[key]

    # 학습 경과 기록
    loss = network.loss(x_batch, t_batch) # 오차를 담는다. 밑에서 사용하지는 않는다.
    train_loss_list.append(loss)

    # 1에폭당 정확도 계산
    if i % iter_per_epoch == 0: # 1에폭돌 때 아래의 코드를 실행해라

```

```

train_acc = network.accuracy(x_train, t_train) # 훈련 데이터의 정확도 출력 60000개
test_acc = network.accuracy(x_test, t_test) # 테스트 데이터의 정확도 출력 10000개
train_acc_list.append(train_acc)
test_acc_list.append(test_acc)
print("train acc, test acc | " + str(train_acc) + ", " + str(test_acc))

```

# 그래프 그리기

```

markers = {'train': 'o', 'test': 's'}
x = np.arange(len(train_acc_list))
plt.plot(x, train_acc_list, label='train acc')
plt.plot(x, test_acc_list, label='test acc', linestyle='--')
plt.xlabel("epochs")
plt.ylabel("accuracy")
plt.ylim(0, 1.0)
plt.legend(loc='lower right')
plt.show()

```

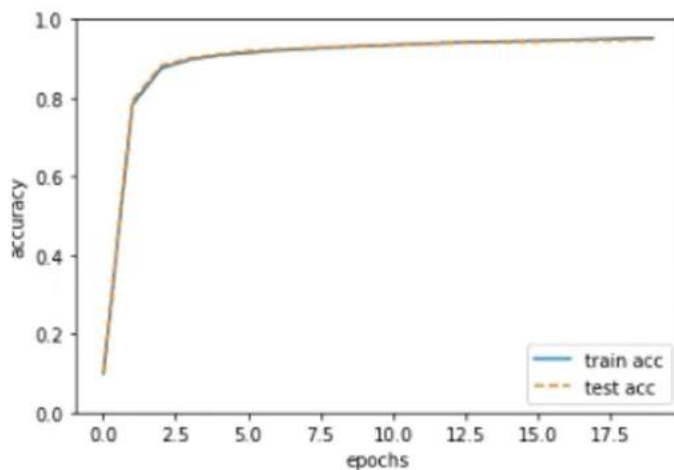
---

신경망이 생성되었습니다.

```

train acc, test acc | 0.0993, 0.1032
train acc, test acc | 0.7824, 0.7927
train acc, test acc | 0.8751, 0.8826
train acc, test acc | 0.8977333333333333, 0.9008
train acc, test acc | 0.9083, 0.9104
train acc, test acc | 0.9144666666666666, 0.9189
train acc, test acc | 0.9205, 0.9228
train acc, test acc | 0.9243833333333333, 0.9267
train acc, test acc | 0.9282166666666667, 0.9296
train acc, test acc | 0.9316166666666666, 0.9321
train acc, test acc | 0.9347166666666666, 0.9352
train acc, test acc | 0.9373833333333333, 0.9372
train acc, test acc | 0.94035, 0.9391
train acc, test acc | 0.9426166666666667, 0.9408
train acc, test acc | 0.9437333333333333, 0.9424
train acc, test acc | 0.9454833333333333, 0.9427
train acc, test acc | 0.9476166666666667, 0.9459
train acc, test acc | 0.94905, 0.9462
train acc, test acc | 0.9507, 0.9466
train acc, test acc | 0.9519, 0.9495

```



## ■ 힘들게 학습 시켜서 만들어 놓은 가중치를 파일로 생성하는 방법

"pickle을 이용한다"

예제1. pickle 파일을 생성하는 예제

```
import pickle
params = [3.1, 4.3, 5.3] # 가중치

with open( "d:\WWweight7.pkl", "wb" ) as f:
    pickle.dump( params, f )
```

**문제89. 4장에서 만든 2층 신경망의 가중치와 바이어스를 pickle 파일로 내리시오!**

코드 맨 아래에 아래의 코드를 넣고 실행한다.

```
import pickle

with open( "d:\WWmnist_weight.pkl", "wb" ) as f:
    pickle.dump( network.params, f )
```

**문제90. 3장에서 사용 3층 신경망 코드를 2층 신경망으로 변경하고 위의 pickle 파일을 셋팅해서 필기체를 분류할 수 있는 신경망을 만드시오!**

```
import pickle

def init_network():
    with open("D:\WWmnist_weight.pkl", "rb") as f:
        network = pickle.load(f)
    return network

network = init_network()
print (network.keys() )
print ( network['W1'].shape) # 2교시 신호 보냈습니다.
print ( network['W2'].shape) # ( , ) 은닉2층의 노드수

import numpy as np
from common.functions import *
from dataset.mnist import load_mnist

# 1. 데이터를 불러옵니다.
(x_train, t_train), (x_test, t_test) = load_mnist(flatten=True, normalize=True,
```

```
one_hot_label=False)
```

```
# 2. 가중치와 바이어스 값을 불러옵니다. (저자가 미리 학습 시킨 가중치와 바이어스)
```

```
network = init_network()
```

```
w1, w2 = network['W1'], network['W2']
```

```
b1, b2 = network['b1'], network['b2']
```

```
# 3. 신경망을 구성합니다.
```

```
# 0층
```

```
x = x_train[0:100] # 일단 10개의 필기체 데이터를 구성합니다.
```

```
# 1층
```

```
y = np.dot(x, w1) + b1
```

```
y_hat = sigmoid(y)
```

```
# 2층
```

```
z = np.dot(y_hat, w2) + b2
```

```
z_hat = sigmoid(z)
```

```
a = np.argmax(z_hat, axis=1) # axis =1 이 축 # 예측값
```

```
b = t_train[0:100] # 실제 정답
```

```
print('총 ', len(a), '중에서 ', sum(a==b), '개 맞추었습니다')
```

## ■ 4장 요약정리

1. 신경망을 학습되게 하려면 무엇이 필요할까 ?

- a. 오차함수 ( 항등함수, 교차엔트로피 )
- b. 수치미분
- c. 미니배치

2. 미분함수를 파이썬으로 구현

3. 편미분함수를 파이썬으로 구현

4. 2층 신경망을 구현( 학습할 수 있는 신경망 )

5. 신경망의 가중치를 pickle 파일로 내림

수치미분으로 신경망을 학습하면 너무 느려서 학습할 수가 없다.

오차 역전파를 이용해서 학습하는 것을 5장에서 배울 예정이다.

## ■ 5장. 오차 역전파 ( p 147 )

수치미분이 너무 느려서 오차 역전파법이라는 것을 설명하는데 오차 역전파법을 쉽게 설명하기 위해 계산그래프 그림을 이용해서 오차역전파법을 설명하고 있다.



↓

시그모이드 함수    소프트맥스 함수    교차엔트로피 함수 ( 분류문제 풀 때 사용 )  
 렐루 함수    항등함수    평균제곱오차 ( 회귀문제 풀 때 사용 )

최종적으로 산출하고자 하는 것은? 오차가 가장 적은 가중치

가중치(  $w_1$  )에 변화가 생겼을 때 오차는 얼마나 달라지는지 알고싶다면?

↓

Ex ) 사과값이 '아주 조금' 올랐을 때 '지불금액'이 얼마나 증가하는지 알고싶다면?

$\partial$ 지불금액

-----

$\partial$  사과값

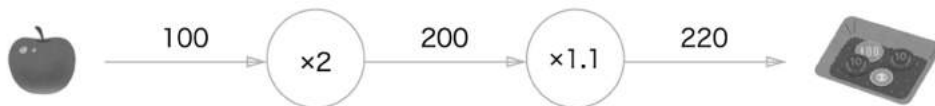
-> 지불금액을 사과값으로 편미분하면 알 수 있다.

계산 그래프의 역전파를 이용하면 위의 계산을 할 수 있다.

**예시 문제1.** 현빈군은 슈퍼에서 1개에 100원인 사과를 2개 샀습니다. 이 때 지불 금액을 구하시오! 단 소비세가 10% 부과됩니다.

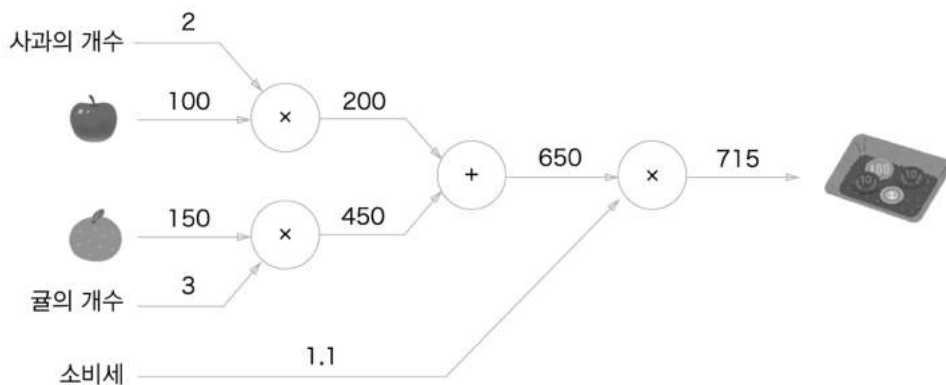
답 :

그림5-1



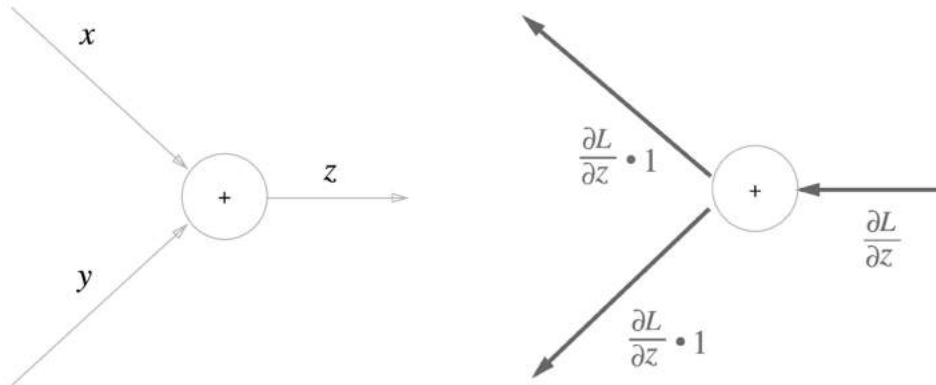
**예시 문제2.** 현빈군은 슈퍼에서 사과를 2개, 귤을 3개 샀습니다. 사과는 1개에 100원, 귤은 1개에 150원 입니다. 소비세가 10%일 때 지불금액을 구하시오!

답 :



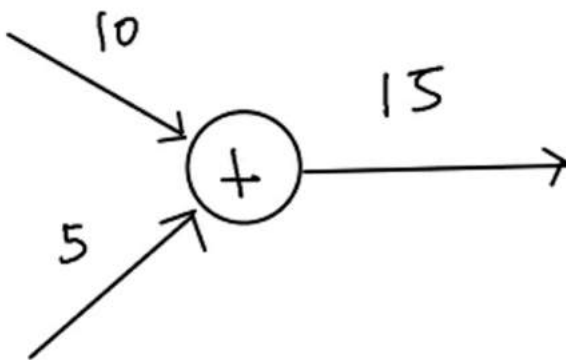
## ■ 덧셈 계산 그래프

그림 5-9

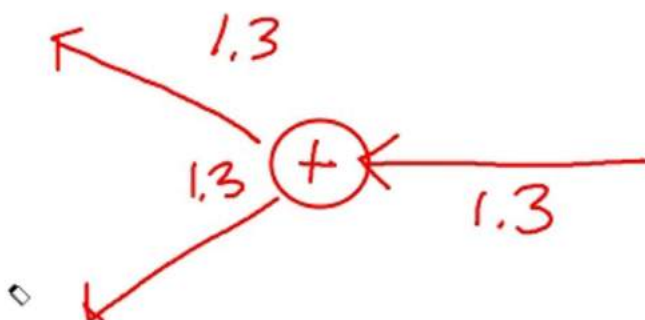


설명 : 덧셈노드 역전파는 상류에서 흘러왔던 값이 그대로 흘러간다.

문제91. 아래의 덧셈 노드 그래프의 순전파 값과 역전파 값을 적으시오!

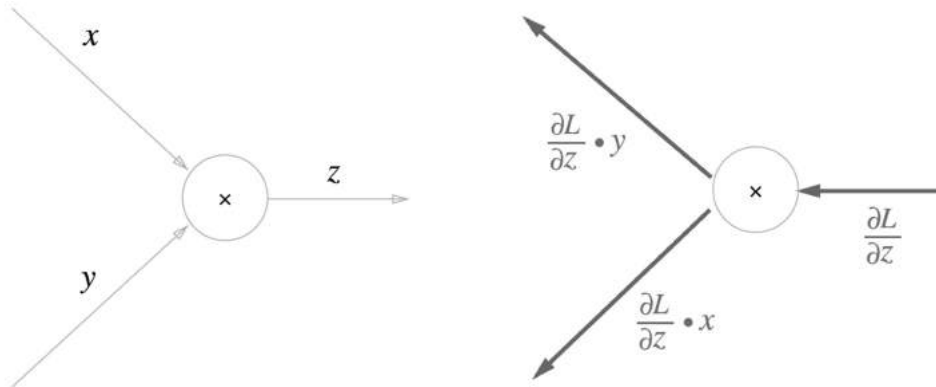


덧셈 : 15

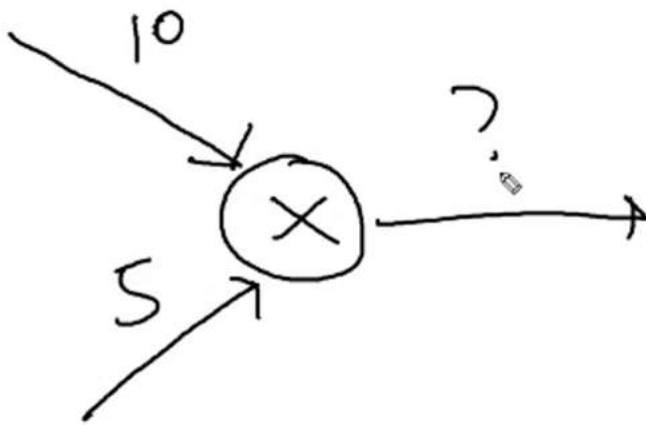


곱셈 : 1.3 , 1.3

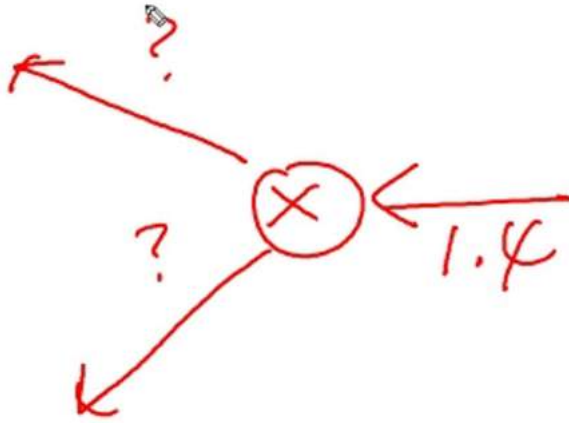
## ■ 곱셈노드의 순전파와 역전파



문제92. 아래의 곱셈 노드의 순전파값과 역전파 값을 각각 적으시오!



답 : 50



답 : 7, 14

### 문제93. 곱셈계층을 파이썬으로 구현하시오! ( p 161 )

```

Class MulLayer:
    def __init__(self):
        self.x = None
        self.y = None

    def forward(self, x, y): #순전파
        self.x = x
        self.y = y
        out = x*y
        return out

    def backward(self, dout): #역전파 그림 5-12
        dx = dout*self.y
        dy = dout*self.x
        return dx, dy

```

### 문제94. 위에서 만든 곱셈 클래스를 객체화 시켜서 아래의 사과가격을 구하시오!

```

apple = 100
apple_num = 2

```

답 :

```

class MulLayer:
    def __init__(self):
        self.x = None
        self.y = None

    def forward(self, x, y): #순전파
        self.x = x
        self.y = y

```

```

out = x*y
return out

```

```

def backward(self, dout): #역전파 그림 5-12
    dx = dout*self.y
    dy = dout*self.x
    return dx, dy

```

```

apple = 100
apple_num = 2

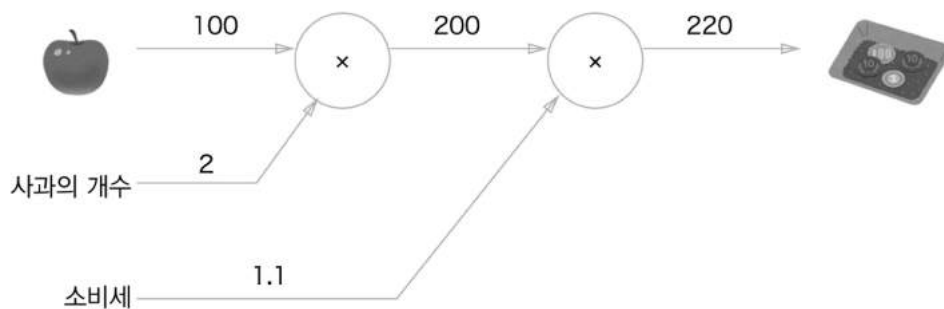
```

```

mul_apple_layer=MulLayer()
apple_price = mul_apple_layer.forward(apple, apple_num)
print(apple_price)

```

**문제95. 곱셈계층 클래스를 객체화 시켜서 아래의 그림 5-2를 계산하시오!**



```

apple = 100
apple_num = 2
tax = 1.1

```

답 :

```

apple = 100
apple_num = 2
tax = 1.1

```

```

mul_apple_layer=MulLayer()
mul_tax_layer=MulLayer()
apple_price = mul_apple_layer.forward(apple, apple_num)
total_price = mul_tax_layer.forward(apple_price, tax)

print(total_price) #220.00000000000003

```

**문제96. 덧셈계층 클래스를 파이썬으로 구현하시오! ( p 163 )**

```

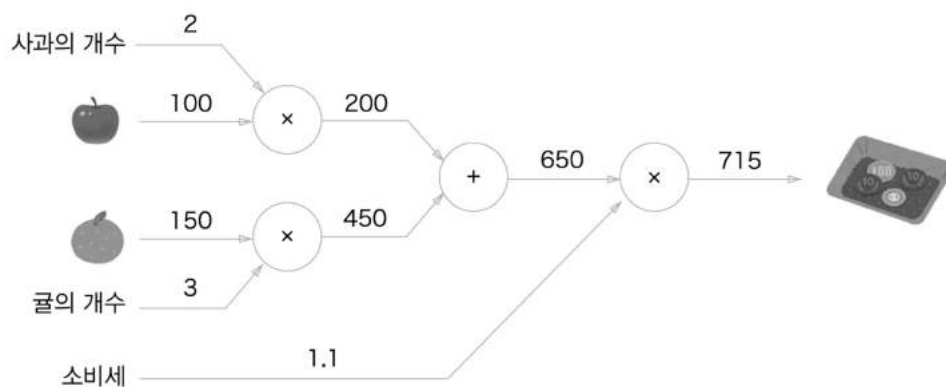
class AddLayer:
    def __init__(self):
        self.x = None
        self.y = None

    def forward(self, x, y):
        self.x = x
        self.y = y
        out = x+y
        return out

    def backward(self, dout):
        dx = dout
        dy = dout
        return dx, dy

```

**문제97. 위에서 만든 곱셈클래스와 덧셈 클래스를 이용해서 그림 5-3의 계산 그래프를 구현하고 최종 값을 계산하시오!**



```

apple = 100
apple_num = 2
orange = 150
orange_num = 3
tax = 1.1

```

```

mul_apple_layer=MulLayer()
mul_orange_layer=MulLayer()
mul_tax_layer=MulLayer()
ao_price_layer = AddLayer()

```

```

apple_price = mul_apple_layer.forward(apple, apple_num)
orange_price = mul_orange_layer.forward(orange, orange_num)

```

```

ao_price = ao_price_layer.forward(apple_price, orange_price)
print(ao_price)

```

```
total_price = mul_tax_layer.forward(ao_price, tax)
```

```
print(total_price) #715.0000000000001
```

## ■ 활성화 함수 계층 구현하기 p 165

### 1. 계산 그래프

- 덧셈 그래프
- 곱셈 그래프
- 렐루 함수 그래프
- 시그모이드 함수 그래프
- 교차엔트로피 함수 그래프( 부록 )
- 소프트맥스 함수 그래프( 부록 )

계산 그래프를 보면서 순전파, 역전파 함수를 생성

## ■ 렐루 함수를 위한 계산그래프 p 165

0보다 큰 값이 입력이 되면 그 값은 그대로 출력하고 0이거나 0보다 작은값이 입력이 되면 0을 출력하는 함수

그림 5-18

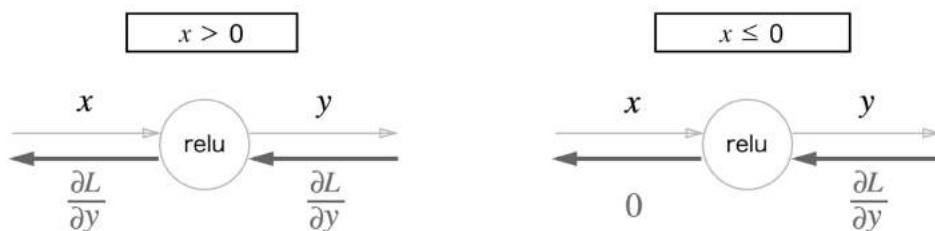
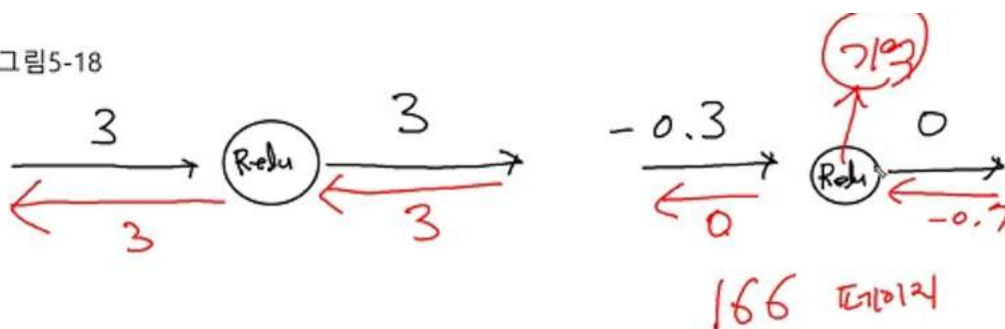


그림5-18



렐루함수를 구현하려면 알아야하는 문법 2가지



1. copy 모듈 사용법

2.  $x[x \leq 0]$ 의 의미

## ■ 1. copy 모듈 사용법

```
a = [ 1, 2, 3 ]
b = a
print( b ) # [1, 2, 3]
a[1] = 6 # a리스트의 1번째 요소를 6으로 변경
print(a) #[1, 6, 3]
print(b) #[1, 6, 3]
```

지금 b는 a와 같은 곳을 바라보고 있는 것인데 이렇게 하지 않고 새롭게 b를 위한 [1, 2, 3]을 만들고 싶다면 copy를 이용해야한다.

```
from copy import copy
```

```
a = [ 1, 2, 3 ]
b = copy(a)
print(a) # [1, 2, 3]
print(b) # [1, 2, 3]
a[1] = 6
print(a) # [1, 6, 3]
print(b) # [1, 2, 3]
```

2.  $x[x \leq 0]$ 의 의미 ( 렐루는 순전파때 보냈던 신호를 기억하고 있어야 하기 때문에 필요함 )  
( 그 자리에 신호가 보내졌었는가를 기억해야한다. )

```
import numpy as np
x = np.array( [[1.0, -0.5], [-2.0, 3.0]] )
print(x)
mask = (x<=0)
print( mask )
```

**결과 :**

```
[[False  True]
 [  True False]]
```

```
out = x.copy()
out[mask] = 0
print( out )
```

**문제98. ( 오늘의 마지막 문제 ) 책 166 페이지의 Relu 클래스를 생성하시오!**

```
x = np.array( [[1.0, -0.5], [-2.0, 3.0]] )
```

```
relu = Relu()  
print( relu.forward(x) )  
dout = np.array( [ [ 2.0, 3.0], [-3.0, 4.0 ] ] )  
print( relu.backward(dout) )
```

03/15

2021년 3월 11일 목요일    오후 4:59

## ■ 딥러닝 수업 복습 ( 밑바닥부터 시작하는 딥러닝 )

- 1장. numpy 사용방법
- 2장. 퍼셉트론
- 3장. 3층 신경망
- 4장. 2층 신경망
- 5장. 오차 역전파를 이용한 2층 신경망
- 6장. 오버피팅과 언더피팅을 막는 방법들
- 7장. cnn
- 8장. 딥러닝의 역사 ( 텐서플로우, 파이토치 )

가지고 있어야 할 딥러닝 기본 포트폴리오 :

- 1. 직접 스크롤링한 사진을 분류하는 신경망 활용 홈페이지
- 2. 영상에서 특정 부분을 detection 하는 것

## ■ sigmoid 계층 ( p 167 )

수학식 : 그림 5.9 ( 책 p167 )

$$y = \frac{1}{1 + \exp(-x)}$$

**예제1. 시그모이드 함수의 순전파를 파이썬 코드로 구현하시오!**

```
import numpy as np
```

```
class Sigmoid:
```

```
    def __init__(self):  
        self.out = None
```

```
    def forward(self,x):  
        out = 1/(1+np.exp(-x))  
        self.out = out
```

**문제99. 위의 Sigmoid 클래스를 객체화 시켜서 아래의 데이터를 forward 함수에 넣고 실행하시오!**

```
x = np.array( [24,32,5] )
```

```
x = np.array( [24,32,5] )  
sigmoid = Sigmoid()  
sigmoid.forward(x) #array([1.      , 1.      , 0.99330715])
```

## **\* 텐서 플로우 2.0 환경 구성**

목차의 8장 : 텐서플로우 2.x 환경 구성 바로가기 참조

<https://cafe.daum.net/oracleoracle/Sedp/287>

인공지능 프로그램 구현할 때:

1. 일반사진을 화가 고흐풍으로 변환하고 싶다( 환경구성 1 )
2. 옥주현 AI를 만들고 싶다. ( 환경구성2 )

두가지 환경을 별도로 분리해서 사용하는게 바람직 함. 이럴때 좋은 방법이 가상환경을 만드는 것이다.

### **■ 텐서플로우 2.0 keras 사용하기**

#### **1. 가상환경 만들기**

```
python --version
```

```
conda create -n snowdeer_env python=3.8.3
```

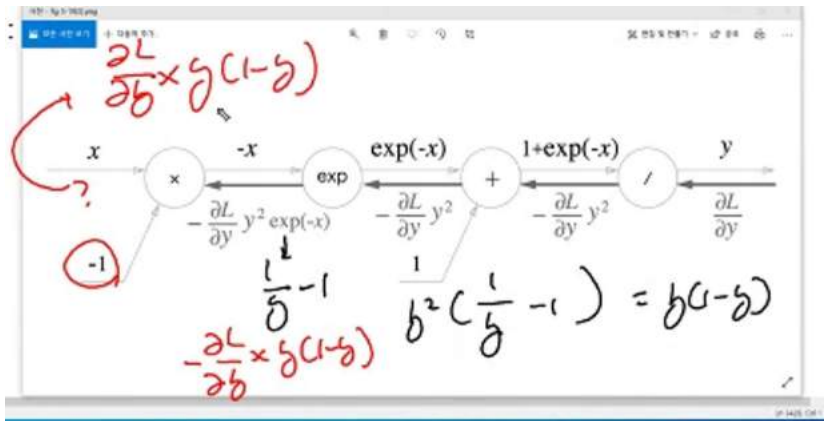
```
import tensorflow as tf  
tf.__version__ #'2.4.1'
```

## **■ 시그모이드 함수를 파이썬으로 날코딩하기( backward부분 구성 )**

책 167~169페이지까지가 시그모이드 함수의 역전파를 구성하는 부분

1단계 : 그림 5-19(1)

$$b = \frac{1}{x} \quad b' = ? \quad x^{-2} \quad b' = -x^{-1-1} = -x^{-2} = -\frac{1}{x^2}$$



**문제100.** 위에서 만든 시그모이드 클래스를 객체화 시켜서 순전파와 역전파를 각각 수행하시오!

```
x = np.array( [[1.0, -0.5], [-2.0, 3.0] ] ) #2x2
sim = Sigmoid()
print( sim.forward(x) )
# 결과 :
[[0.73105858 0.37754067]
 [0.11920292 0.95257413]]
```

# 순전파일때는 2X2행렬로 흘러갔으면 역전파일때도 2x2행렬로 흘러온다.  
 # 순전파일때는 입력값이 흘러가고 시그모이드를 통과하면 확률로 출력이 된다.  
 # 역전파일때는 오차가 흘러온다. 오차가 흘러오면서 미분대신 만든 도함수식과 오차가 같이  
 # 계산되면서 오차가 계속 흘러오고 최종적으로 가중치에 오차가 반영이 되면서 가중치가 갱신  
 # 되어진다.

```
dout = np.array( [ [2.0, 3.0], [-3.0, -4.0] ] )
print( sim.backward( dout ) )
```

```
#결과 :
[[ 0.39322387  0.70501114]
 [-0.31498076 -0.18070664]]
```

## ■ 파이썬으로 직접 sigmoid 함수를 만들고 데이터를 넣고 계산되게 했는데 지금부터는 텐서플로를 이용해 sigmoid 함수에 데이터를 넣고 계산되게 해보기

### 파이썬 날코딩 vs 텐서플로우

파이썬 날코딩 : 우리가 직접 sigmoid 함수를 생성

텐서플로우 : 구글의 어떤 개발자가 sigmoid 함수 생성

```
import tensorflow as tf
```

```
x = tf.constant( [24.0, 32.0, 5.0], dtype = tf.float32) # 3개의 상수값을 생성함 ( 실수형으로 생성해야함)
```

```
tf.math.sigmoid(x) #위의 3개의 숫자를 시그모이드에 흘려보낸다.
```

문제101.( 점심시간 문제 ) 아래의 2x2행렬을 텐서 플로우의 시그모이드에 흘려 보내시오

```
x = np.array( [[1.0, -0.5], [-2.0, 3.0] ] )
```

```
x = tf.constant( [[1.0, -0.5], [-2.0, 3.0] ] )
```

```
tf.math.sigmoid(x)
```

## ■ 5장의 오차 역전파를 구현할 파이썬 코드 구현

1. relu 함수
2. sigmoid 함수
3. Affine 계층
4. softmax 함수
5. 오차 함수

### ■ Affine 계층 p 170

신경망의 순전파때 수행하는 행렬의 내적을 기하학에서는 어파인 변환이라고 한다.

그래서 신경망에서 입력값과 가중치의 내적의 합에 바이어스를 더하는 층을 Affine 계층이라고 해서 구현한다.

" 지금까지의 계산 그래프는 노드 사이에 '스칼라값'이 흘렀는데 이에 반해 이번에는 '행렬'이 흐르고 있어서 Affine 계층 구현이 필요하다 "

\* 어파인 계층을 이해하기 위한 행렬의 종류 설명 :

1. 전치행렬
2. 역행렬

**문제102. p 175에 나오는 Affine 계층 클래스를 생성하시오!**

```
import numpy as np
class Affine:
    def __init__(self, W, b):
        self.W = W
        self.b = b
        self.x = None
        self.dW = None
        self.db = None

    def forward(self,x):
        self.x = x
        out = np.dot(x, self.W) + self.b
        return out

x = np.array( [1,2] )
W = np.array( [ [1,3,5], [2,4,6] ] )
b = np.array([1, 1, 1])

affine1 = Affine(W,b)
affine1.forward(x) # array( [ 6, 12, 18 ] )
```

※ 순전파는 위와 같이 구현하고 역전파를 계산그래프를 그려서 도함수를 확인해서 도함수로 backward 함수를 구현하면 된다.

**문제103. 아래에 나오는 batch로 데이터를 신경망에 입력했을 때의 순전파를 구현하시오!**

```
x = np.array( [[1,2], [3,4]] )
W = np.array( [ [1,3,5], [2,4,6] ] )
b = np.array([1, 1, 1])

affine2 = Affine(W,b)
```

```
affine2.forward(x)
```

결과 :

```
array([[ 6, 12, 18],  
       [12, 26, 40]])
```

## ■ Affine 계층의 역전파 계산 그래프

```
import numpy as np  
class Affine:  
    def __init__(self, W, b):  
        self.W = W  
        self.b = b  
        self.x = None  
        self.dW = None  
        self.db = None  
  
    def forward(self, x):  
        self.x = x  
        out = np.dot(x, self.W) + self.b  
        return out  
  
    def backward(self, dout):  
        dx = np.dout(dout, self.W.T)  
        self.dW = np.dot(self.x.T, dout)  
        self.db = np.sum(dout, axis = 0)  
        return dx
```

순전파에서는 broadcast해주고 역전파때는 sum ( 행끼리 합해줌 )을 해준다. --> 곱을 해주기 위해

※ Affine 계층의 역할은 신경망에서 층을 하나 구성하는 부분이다.

**책 182 페이지의 코드를 보면 Affine 계층 활용하는 코드:**

```
self.layers['Affine'] = Affine( self.params['W1'], self.params['b1'] )
```

**아래의 코드와 비교해 보기**

```
affine1 = Affine( W, b )  
affine1.forward(x)
```



## ■ 텐서플로우 2.x 버전으로 퍼셉트론 구현하기

"인공 신경 세포 하나를 구현 "

\* 퍼셉트론에서 쓰이는 입력데이터와 타겟 4가지

1. and 게이트
2. or 게이트
3. not and 게이트      단층으로 구현
- 
4. xor 게이트      다층으로 구현

## ■ and 게이트 구현

```
import tensorflow as tf
tf.random.set_seed(777) # 시드를 설정한다.
```

```
import numpy as np
from tensorflow.keras.models import Sequential # 신경망 모델 구성
from tensorflow.keras.layers import Dense # 완전 연결계층 ( Affine 계층 부분 )
from tensorflow.keras.optimizers import SGD # 경사감소법 ( 6장의 내용 )
from tensorflow.keras.losses import mse # 오차함수
```

### # 데이터 준비

```
x = np.array( [ [0, 0], [1, 0], [0, 1], [1, 1] ] ) # 입력값
y = np.array( [ [0], [0], [0], [1] ] ) # 라벨
```

### #모델 구성하기 ( 신경망 구성 시작하겠다 )

```
model = Sequential()
```

### #단층 퍼셉트론 구현하기

```
model.add( Dense( 1, input_shape =( 2, ), activation ='linear') )
```

### # 모델 준비하기

```
model.compile( optimizer= SGD(), # 경사감소법의 종류를 기술( 잘 모르면 adam 쓰면 됨 )
               loss= mse, # 오차함수( MSE(회귀), cross_entropy_error(분류) )
               metrics = ['acc'] ) # list 형태로 평가지표를 전달한다. 'acc' = 정확도기준
               # 정확도가 높아지는 것을 목표로 학습해달라는 뜻
```

### # 학습 시키기

```
model.fit(x, y, epochs = 500) # 입력 데이터와 정답을 입력하면서 500에폭 돌면서 학습 시키겠다.
```

**문제104. AND 게이트 퍼셉트론을 구현한 신경망에서 만들어낸 가중치를 출력하시오 !**

```
print (model.get_weights())
```

**결과 :**

```
[array([[0.45816228],  
       [0.3393745 ]], dtype=float32), array([-0.12992516], dtype=float32)]
```

**문제105. 위의 and 게이트 퍼셉트론 신경망이 예측한 결과를 출력하시오!**

```
model.evaluate(x , y ) # 모델을 평가한다.
```

**결과 :**

```
[0.06974281370639801, 1.0] # [오차, 정확도]
```

```
result = model.predict(x) # 입력값 넣어서 예측값을 출력  
print( result.round( ) )
```

**결과 :**

```
[[ -0.]  
 [  0.]  
 [  0.]  
 [  1.]]
```

```
# 학습시키기
```

```
model.fit(data, label, epochs = 500)
```

```
model.evaluate(data, label)
```

```
result = model.predict(data)
```

```
for i in result:
```

```
    for j in i:
```

```
        print (round(j))
```

```
0
```

```
0
```

```
0
```

```
1
```

**문제106. or 게이트를 구현하시오 !**

```
import tensorflow as tf  
import numpy as np
```

```

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.losses import mse

tf.random.set_seed(777)

# 데이터 준비하기
data = np.array([[0, 0], [1, 0], [0, 1], [1, 1]])
label = np.array([[0], [1], [1], [1]])

# 모델 구성하기
model = Sequential()
model.add(Dense(1, input_shape = (2, ), activation = 'linear')) # 단층 퍼셉트론을 구성합니다

# 모델 준비하기
model.compile(optimizer = SGD(), loss = mse, metrics = ['acc']) # list 형태로 평가지표를 전달합니다

# 학습시키기
model.fit(data, label, epochs = 500)

model.evaluate(data, label)

result = model.predict(data)
for i in result:
    for j in i:
        print (round(j))

0
1
1
1

```

## 문제107. Not and 게이트를 구현하시오 !

```

import tensorflow as tf
import numpy as np

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.losses import mse

tf.random.set_seed(777)

# 데이터 준비하기
data = np.array([[0, 0], [1, 0], [0, 1], [1, 1]])
label = np.array([[1], [1], [1], [0]])

```

```

# 모델 구성하기
model = Sequential()
model.add(Dense(1, input_shape = (2, ), activation = 'linear')) # 단층 퍼셉트론을 구성합니다

# 모델 준비하기
model.compile(optimizer = SGD(), loss = mse, metrics = ['acc']) # list 형태로 평가지표를 전달합니다

# 학습시키기
model.fit(data, label, epochs = 500)

model.evaluate(data, label)

result = model.predict(data)
for i in result:
    for j in i:
        print (round(j))

```

## 문제108. Xor 게이트를 구현하시오 !

**힌트: 다층으로 구성해야한다 !!**

단층으로 아래의 데이터를 넣으면 0 1 0 1 이렇게 출력한다.

```

# 데이터 준비하기
data = np.array([[0, 0], [1, 0], [0, 1], [1, 1]])
label = np.array([[0], [1], [1], [0]])

답:
import tensorflow as tf
import numpy as np

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.losses import mse

tf.random.set_seed(777)

# 데이터 준비하기
data = np.array([[0, 0], [1, 0], [0, 1], [1, 1]])
label = np.array([[0], [1], [1], [0]])

# 모델 구성하기
model = Sequential()
model.add(Dense(32, input_shape = (2, ), activation = 'relu')) # 1층 구현

```

※ 설명: 항상 모델의 첫번째 층은 데이터의 형태(위의 코드에서 input\_shape 인자)를 전달해줘야

한다는 점을 기억해야합니다.

```
model.add(Dense( 1, activation='sigmoid') ) # 2층 출력층 구현
```

# 모델 준비하기

```
model.compile(optimizer = RMSprop(), loss = mse, metrics = ['acc']) # list 형태로 평가지표를 전달합니다
```

※ 설명: 모델을 model.add 로 구성했으면 컴파일(compile) 함수를 호출해서 학습과정을 설정합니다.

※ optimizer에는 경사하강하는 방법을 써준다.

종류 ? SGD(), RMSprop(), Adam() 등이 있다.

※ metrics = ['acc'] --> 학습과정을 모니터링 하기 위해 설정

# 평균제곱오차 회귀문제

```
model.compile(optimizer = RMSprop(), loss = mse, metrics = ['acc'])
```

# 이항분류 문제 ( 이파리나 개고양이와 같은 2가지 분류)

```
model.compile(optimizer = RMSprop(), loss = binary_crossentropy, metrics = ['acc'])
```

# 다항 분류 문제

```
model.compile(optimizer = RMSprop(), loss = categorical_crossentropy, metrics = ['acc'])
```

※ 옵티마이저의 종류 : SGD(), RMSProp(), Adam(), NAadm() 등이 있다.

※ 손실 함수(오차함수)의 종류: mse ( mean squared\_error) 와  
binary\_crossentropy, categorical\_crossentropy 등이 있다.

※ metrics = ['acc'] : 학습과정을 모니터링 하기 위해서 설정합니다.

acc 와 같이 문자열로 지정해서 전달하면 됩니다

# 학습시키기

```
model.fit(data, label, epochs = 100)
```

※ 설명:

```
model.fit(data, label, epochs = 100)
```

```
model.fit(data, label, epochs = 100, validation_data=(val_data, val_label) )
```

validation\_data는 검정 데이터인데 모델의 성능을 모니터링 하기 위해서 사용한다. 검정 데이터는 훈련 데이터를 일부를 가지고 만든 데이터로 훈련이 잘되는지 성능을 보기위한 평가지표로 사용한다.

```
model.evaluate(data, label)
```

※ 설명: evaluate() 함수를 사용하면 손실과 평가 지표에 대한 정보를 확인할 수 있다.

결과의 예: [ 0.210610177, 1.0 ]

```
result = model.predict(data)
```

```
for i in result:  
    for j in i:  
        print (round(j))
```

```
0  
1  
1  
0
```

## ■ 밑바닥 딥러닝의 5장의 mnist 신경망 ( 필기체 분류 신경망 )을 텐서플로우 2.0으로 구현

### 3층 신경망 구현

#### \* 큰 순서

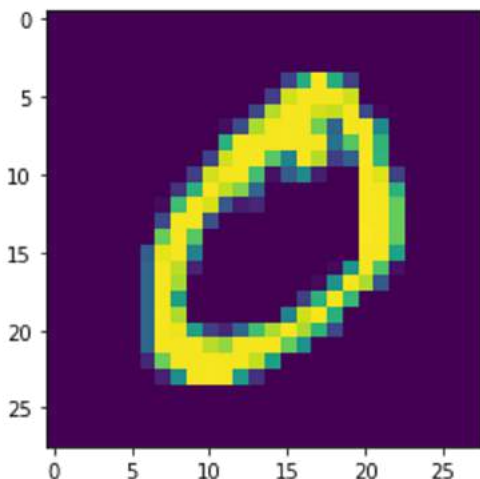
#### # 1. mnist 데이터를 불러온다.

```
from tensorflow.keras.datasets.mnist import load_data  
(x_train, y_train), ( x_test, y_test ), = load_data(path='mnist.npz')  
print( x_train.shape, y_train.shape) # (60000, 28, 28) (60000,)
```

#### # 숫자 하나를 시각화 해본다.

```
import matplotlib.pyplot as plt
```

```
img = x_train[1,]  
plt.figure()  
plt.imshow(img)
```



#### # 2. 훈련데이터를 훈련과정에서의 검증을 위해 훈련데이터의 일부를 검증데이터로 구성한다.

**# ( 훈련데이터 0.7, 검증데이터 0.3 으로 구성 )**

**# 60000장을 7:3으로 나눈다.**

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_val, y_train, y_val = train_test_split( x_train, y_train, test_size = 0.3, random_state=777 )
print( x_train.shape ) # (42000, 28, 28)
print( x_val.shape ) # (18000, 28, 28)
print( x_test.shape ) # (10000, 28, 28)
```

**# 3. 모델에 데이터를 입력하기 전 정규화를 한다. ( 데이터 전처리 )**

신경망은 입력 데이터의 스케일에 매우 민감하므로 적절한 전처리가 필수이다. 숫자의 하나의 픽셀이 0~255 사이의 범위에 있기 때문에 하나의 픽셀을 255로 나누면 0~1사이의 숫자로 스케일이 된다.

```
num_x_train = x_train.shape[0]
num_x_val = x_val.shape[0]
num_x_test = x_test.shape[0]

x_train = x_train.reshape( (num_x_train, 28 * 28))/255
x_val = x_val.reshape( (num_x_val, 28 * 28))/255
x_test = x_test.reshape( (num_x_test, 28 * 28))/255

print(x_train.shape) # (42000, 784)
print(x_val.shape) # (18000, 784)
print(x_test.shape) # (10000, 784)
```

**# 4. 모델에 입력할 정답을 one hot 인코딩 한다.**

```
from tensorflow.keras.utils import to_categorical
y_train = to_categorical(y_train)
y_val = to_categorical(y_val)
y_test = to_categorical(y_test)
print(y_test)
```

**결과 :**

```
[[0. 0. 0. ... 1. 0. 0.]
 [0. 0. 1. ... 0. 0. 0.]
 [0. 1. 0. ... 0. 0. 0.]
 ...
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]
 [0. 0. 0. ... 0. 0. 0.]]
```

**# 5. 모델을 구성한다. ( 3층 신경망으로 구성 )**

```

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential()
model.add( Dense( 100, activation = 'relu', input_shape=(784, ) ) ) # 1층 구성
model.add( Dense( 50, activation = 'relu' ) ) #2층 구성
model.add( Dense( 10, activation='softmax' ) )

```

#### # 6. 모델 과정을 설정한다.

```

from tensorflow.keras.losses import categorical_crossentropy

model.compile(optimizer = 'adam',
              loss = 'categorical_crossentropy',
              metrics = ['acc'])

```

※ 기본 학습률( 러닝 레이트 ) 0.001로 설정되어있다.

#### # 7. 모델을 학습시킨다.

```

history = model.fit ( x_train, y_train, epochs = 30, batch_size = 100, validation_data=(x_val,
y_val) )

```

#### # 8. 테스트 데이터를 넣고 예측해 본다.

```

result = model.predict( x_test )
print( result )

```

결과 :

```

[[5.0352647e-12 1.1290672e-14 2.7705332e-10 ... 1.0000000e+00
 1.6616508e-13 5.8167257e-13]
 [2.1967891e-11 1.5134864e-10 9.9999988e-01 ... 5.1350154e-20
 1.2354542e-08 3.3273435e-24]
 [5.5216431e-09 9.9999321e-01 9.2114405e-08 ... 3.0020410e-06
 1.9415361e-06 3.3162684e-11]
 ...
 [2.8209559e-21 2.3480624e-14 3.5324939e-24 ... 4.6226705e-13
 8.4720988e-18 6.1241311e-11]
 [2.0866499e-18 1.3682803e-19 1.3692002e-25 ... 7.0658311e-19
 4.6416599e-08 9.2415546e-28]
 [2.2681929e-18 1.8895064e-20 2.4008727e-21 ... 8.6400508e-26
 4.9192580e-17 6.1200125e-25]]

```

#### # 9. 확률벡터 10000개에 numpy argmax를 이용해서 최대값의 인덱스를 뽑아낸다.

```

import numpy as np

```



```
np.argmax( results, axis = 1 )
```

결과 :

```
array([7, 2, 1, ..., 4, 5, 6], dtype=int64)
```

# 10. ( 오늘의 마지막 문제 ) 위에 출력된 예측 숫자들 10000개와 테스트 데이터 숫자10000개와 비교해서 정확도가 어떻게 되는지 출력하시오 !

```
import numpy as np
```

```
y_hat = np.argmax( results, axis = 1 )
```

```
y_actual = np.argmax( y_test, axis=1)
```

```
sum((y_hat==y_actual)/10000) # 0.97659999999999088
```

# 03/16

2021년 3월 16일 화요일 오전 9:44

## ■ 딥러닝 수업 복습

1장. numpy 사용법

2장. 퍼셉트론

3장. 3층 신경망 구성 ( 학습 x ) --> 저자가 만들어 온 가중치로 바로 셋팅해서 신경망 구성

4장. 2층 신경망 구성 ( 학습 o ) --> 수치미분을 이용해서 학습시킴

5장. 오차 역전파를 이용해서 2층 신경망 구성 ( 학습 o ) --> 오차 역전파를 이용해서 학습시킴

파이썬으로 다 날 코딩해서 신경망 구성(4~5년 책)---> 그림이 많고 설명이 자세함

수업	vs	현업
이론과 개념 정리(책) 파이썬		tensorflow pytorch

## ■ 5장 복습

수치미분이 너무 느리기때문에 오차 역전파를 사용해서 기울기를 구해서 가중치를 갱신해줘야 한다. 오차 역전파를 이해하기 쉽도록 그림으로 코딩 전에 이론을 이해하는 과정이 계산 그래프이다.

" 그림 --> 코딩 "

1. 계산 그래프를 통해서 덧셈그래프와 곱셈 그래프를 이해
2. Relu 함수 클래스 생성
3. sigmoid 함수 계산 그래프, 클래스 생성( 순전파, 역전파 )
4. Affine 계층 ( 행렬의 내적 ) 계산 그래프, 클래스 생성
5. 텐서플로우 2.x으로 3층 신경망을 구현

**문제109. 지금만든 3층 신경망의 활성화 함수를 relu로 했을때의 테스트 데이터의 정확도는 0.976이었다. sigmoid 함수로했을때는 정확도가어떻게 되는지 확인 하시오!**

```
from tensorflow.keras.datasets.mnist import load_data
(x_train, y_train), ( x_test, y_test ), = load_data(path='mnist.npz')
print( x_train.shape, y_train.shape) # (60000, 28, 28) (60000,)
```

```
print(y_train)
```

```
# 숫자 하나를 시각화 한다.
```

```
import matplotlib.pyplot as plt
```

```
img = x_train[1,]
plt.figure()
plt.imshow(img)
```

```
#2. mnist를 진행한다.
```

```
from sklearn.model_selection import train_test_split
```

```
x_train, x_val, y_train, y_val = train_test_split( x_train, y_train, test_size = 0.3, random_state=
777 )
```

```
print( x_train.shape ) # (42000, 28, 28)
```

```
print( x_val.shape ) # (12600, 28, 28)
```

```
print( x_test.shape ) # (10000, 28, 28)
```

```
# 3. 정규화를 진행한다
```

```
num_x_train = x_train.shape[0]
```

```
num_x_val = x_val.shape[0]
```

```
num_x_test = x_test.shape[0]
```

```
x_train = x_train.reshape( (num_x_train, 28 * 28))/255
```

```
x_val = x_val.reshape( (num_x_val, 28 * 28))/255
```

```
x_test = x_test.reshape( (num_x_test, 28 * 28))/255
```

```
print(x_train.shape)
```

```
print(x_val.shape)
```

```
print(x_test.shape)
```

```
# 정답 데이터를 onehot encoding한다.
```

```
from tensorflow.keras.utils import to_categorical
```

```
y_train = to_categorical(y_train)
```

```
y_val = to_categorical(y_val)
```

```
y_test = to_categorical(y_test)
```

```
print(y_test)
```

```
# 5. 신경망 모델을 구성한다. ( 3층 신경망으로 구성 )
```

```
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Dense
```

```
model = Sequential()
```

```
model.add( Dense( 100, activation = 'sigmoid', input_shape=(784, ) ) ) # 1층 구성
```

```
model.add( Dense( 50, activation = 'sigmoid' ) ) #2층 구성
```

```
model.add( Dense( 10, activation='softmax' ) )
```

# 6. 신경망 모델 과정을 설정한다.

```
from tensorflow.keras.losses import categorical_crossentropy
```

```
model.compile(optimizer = 'adam',  
              loss = 'categorical_crossentropy',  
              metrics = ['acc'])
```

# 7. 신경망 모델을 훈련시킨다.

```
history = model.fit ( x_train, y_train, epochs = 30,  
                     batch_size = 100,  
                     validation_data=(x_val, y_val) )
```

# 8. 모델을 평가한다.

```
model.evaluate(x_test, y_test )
```

# 9. 테스트 데이터의 정확도를 확인한다.

```
results = model.predict( x_test )  
print( results )
```

# 정확도 확인

```
import numpy as np
```

```
y_hat = np.argmax( results, axis = 1 )
```

```
y_label = np.argmax( y_test, axis=1)
```

```
sum((y_hat==y_label)/len(y_label))
```

## 문제110. 위의 결과를 시각화해서 성능을 확인하시오!

# 시각화

```
import matplotlib.pyplot as plt
```

```
his_dict = history.history
```

```
loss = his_dict['loss']
```

```
val_loss = his_dict['val_loss'] # 검증 데이터가 있는 경우 'val_' 수식어가 붙습니다.
```

```
epochs = range(1, len(loss) + 1)
```

```
fig = plt.figure(figsize = (10, 5))
```

# 훈련 및 검증 손실 그리기

```
ax1 = fig.add_subplot(1, 2, 1)
```

```
ax1.plot(epochs, loss, color = 'blue', label = 'train_loss')
```

```
ax1.plot(epochs, val_loss, color = 'orange', label = 'val_loss')
```

```

ax1.set_title('train and val loss')
ax1.set_xlabel('epochs')
ax1.set_ylabel('loss')
ax1.legend()

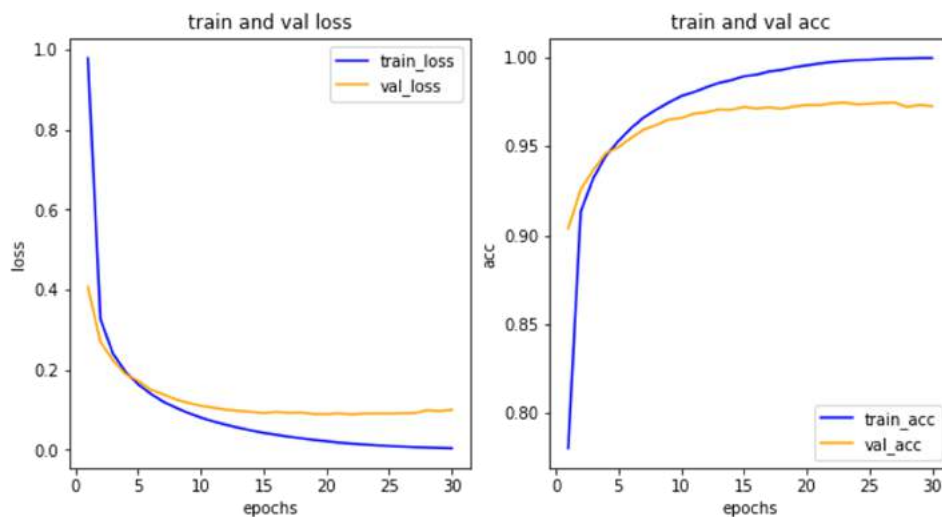
acc = his_dict['acc']
val_acc = his_dict['val_acc']

# 훈련 및 검증 정확도 그리기
ax2 = fig.add_subplot(1, 2, 2)
ax2.plot(epochs, acc, color = 'blue', label = 'train_acc')
ax2.plot(epochs, val_acc, color = 'orange', label = 'val_acc')
ax2.set_title('train and val acc')
ax2.set_xlabel('epochs')
ax2.set_ylabel('acc')
ax2.legend()

plt.show()

```

시각화를 통해 오버피팅이 일어나는 것이 확인 가능하다.



검증데이터와 훈련데이터 사이의 간격이 넓다 --> 오버피팅

## 문제111. 위의 3층 신경망을 4층 신경망으로 구현하시오!

# 5. 신경망 모델을 구성한다. ( 3층 신경망으로 구성 )

```

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential()
model.add( Dense( 100, activation = 'sigmoid', input_shape=(784, ) ) )#1층구성
model.add( Dense( 50, activation = 'sigmoid' ) ) #2층구성
model.add( Dense( 25, activation = 'sigmoid' ) ) #3층 구성

```

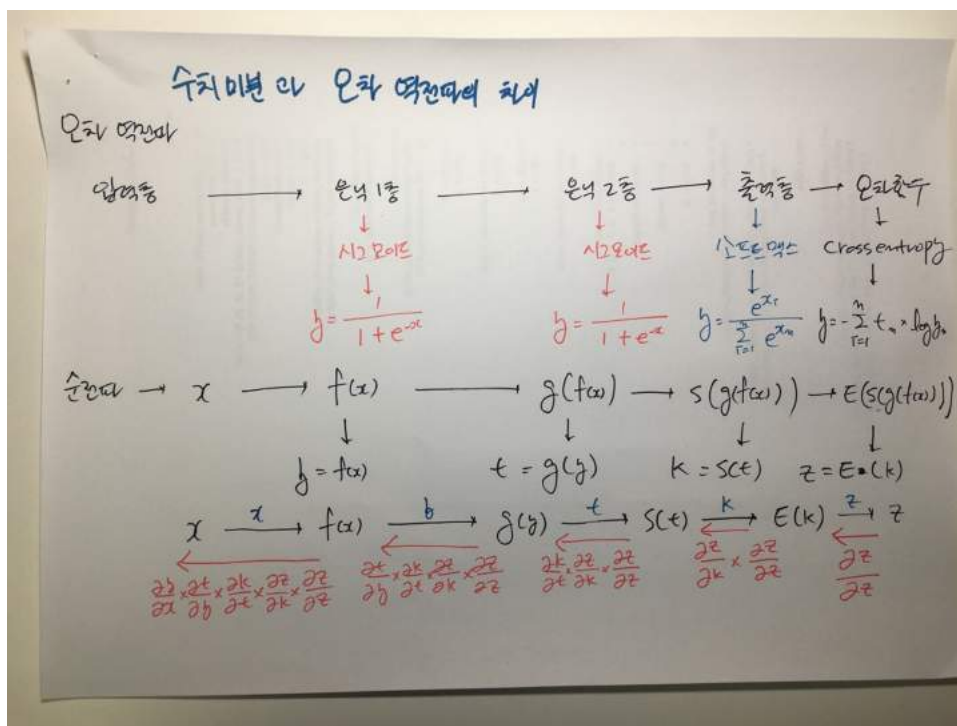
model.add( Dense( 10, activation='softmax' ) )

4층으로 변경했더니 3층일때 0.976에서 0.971로 정확도가 더 줄어든다.

■ 텐서플로우가 아니라 파이썬으로 3층 신경망을 구현한다면 어떻게 될까?

■ 4장의 수치미분과 5장의 오차 역전파의 차이가 무엇인가?

( 연쇄법칙 p153 )



합성함수 : 여러함수로 구성된 함수

- 수치미분은 오차함수를 바로 가중치로 편미분해서 기울기를 구했지만 이렇게 계산하면 사람도 어렵고 컴퓨터도 어렵다. 할 수는 있지만 시간이 많이 걸린다. 그래서 합성함수 미분처럼 연쇄법칙에 의해서 신경망안에 들어가는 각각의 함수들의 도함수를 구해서 도함수를 backward에 넣고 기울기가 역전파 되어서 흘러가게 하면 그 기울기로 가중치를 갱신하는것이 수치미분으로 했을때 보다 훨씬 빠르다.

5장 182~183 페이지가 그 전체코드다.

5장의 전체코드를 텐서플로우, 파이토치로 구현할 수 있어야 한다.

## ■ 6장. 학습관련 기술들( 인공신경망의 언더피팅과 오버피팅을 막는 방법 )

### 1. underfitting을 방지하는 방법

- 고급 경사하강법의 종류
- 가중치 초기값 설정
- 배치 정규화

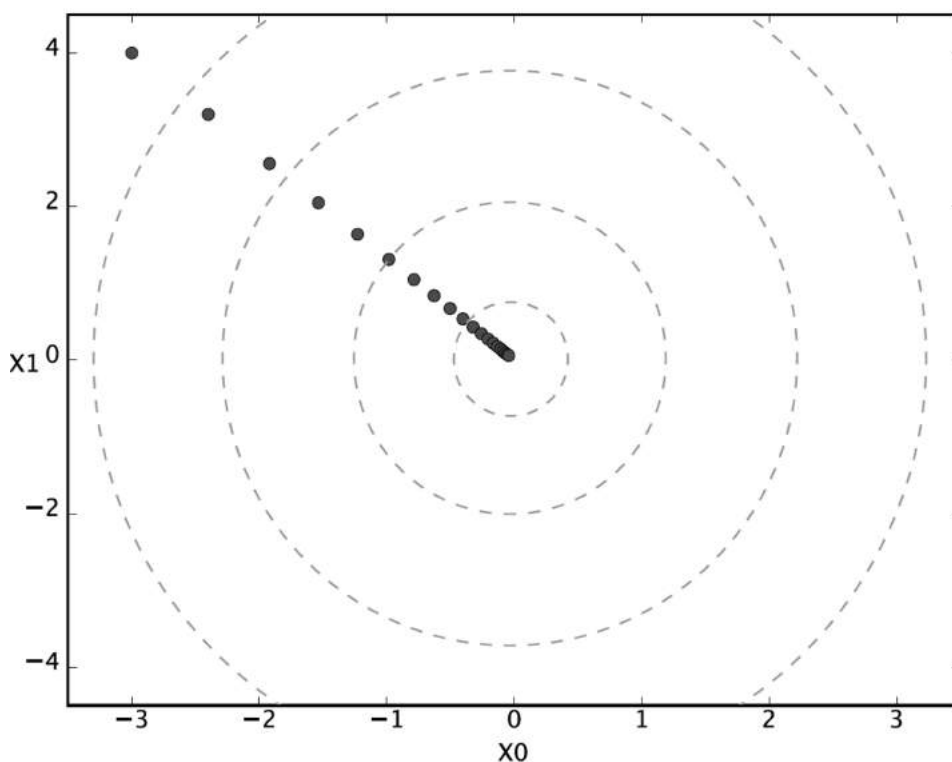
### 2. overfitting을 방지하는 방법

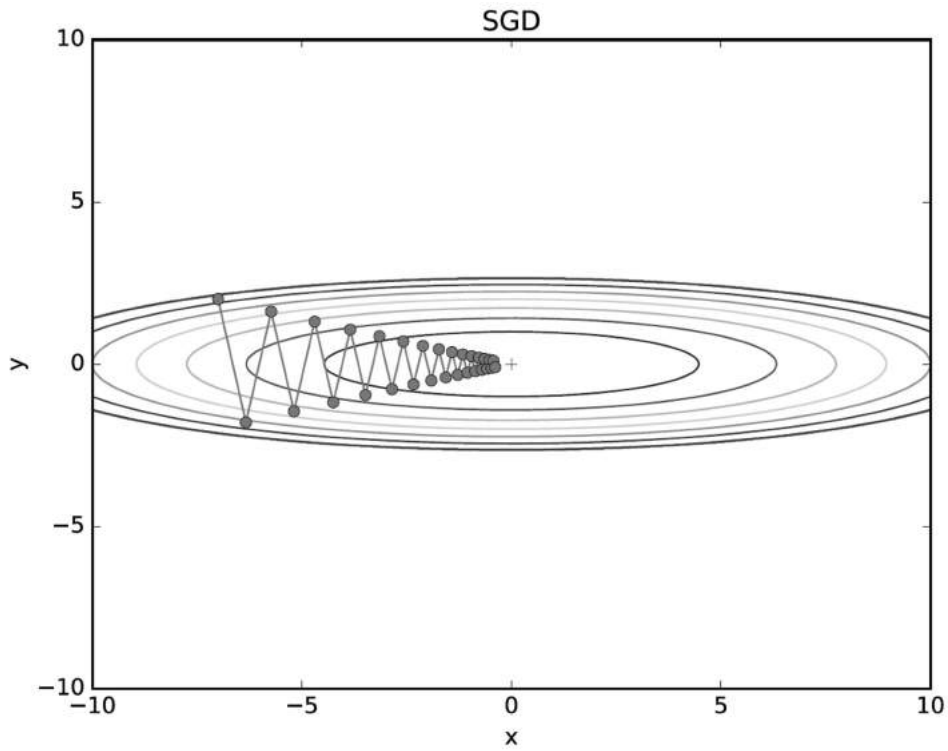
- 드롭아웃
- 엘리스탑( keras의 기능 )

## ■ 고급 경사하강법 간단하게 정의

### 1. GD( Gradient Descent ) : 학습 데이터를 다 입력하고 한걸음 이동하는 경사하강법

그림 4-10, 그림 6-3





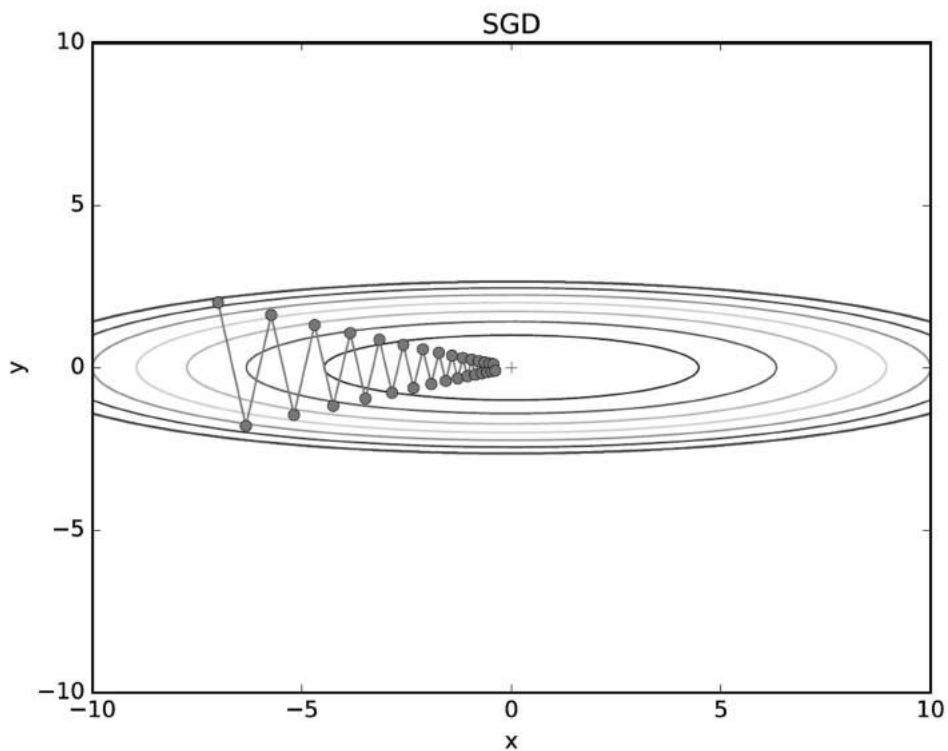
단점 : 학습데이터를 신경망에 다 입력해서 한걸음 이동하므로 시간이 많이 걸린다.

## 2. SGD ( Stochastic Gradient Descent )

GD의 단점을 개선한 경사 하강법이다.

Stochastic( 확률적 ) 경사하강법으로 복원추출 미니배치해서 경사가 기울어진 방향으로 학습해 나가는 방법이다.

그림 6-3





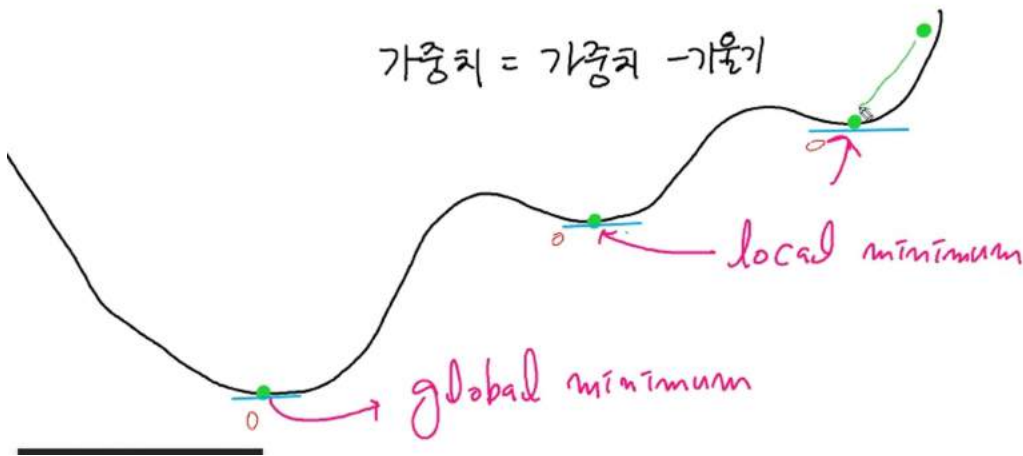
문제112. ( 점심시간 문제 ) 5장 전체코드( 텐서 플로우 2.0으로 구현 )의 경사하강법을 SGD로 구현해서 수행하고 테스트 데이터 정확도와 시각화를 올리시오!

SGD + Relu : 0.9524

SGD의 단점 : local minimum에 잘 빠진다.

극복방법 :

**3. Momentum** - 관성을 이용해 local minima에서 빠져나가게끔 경사하강하는 방법



$$\mathbf{v} \leftarrow \alpha \mathbf{v} - \eta \frac{\partial L}{\partial \mathbf{W}}$$

모멘텀은 tensorflow2.0 에는 따로 없지만 Adam과 같은 다른 경사하강법 종류에 모멘텀의 장점이 구현이 되어져있다.

**4. Adagrade** : learning rate( 학습률 )를 자동조절 되게 하는 경사하강법. 처음에는 학습률을 크게 하고 나중에는 학습률을 작게하여 점차 줄여가는 경사하강법으로 각각의 매개변수( w1, w2, w3 )에 맞춤형 값을 만들어준다.

$$\mathbf{h} \leftarrow \mathbf{h} + \frac{\partial L}{\partial \mathbf{W}} \odot \frac{\partial L}{\partial \mathbf{W}}$$

$$\mathbf{W} \leftarrow \mathbf{W} - \eta \frac{1}{\sqrt{\mathbf{h}}} \frac{\partial L}{\partial \mathbf{W}}$$

**문제113. Adagrade로 3층 신경망의 경사하강법을 변경하고 정확도를 확인하십시오. SGD일때는 정확도가 아래와 같습니다.**

**SGD + relu : 0.9524**

**# 6. 신경망 모델 과정을 설정한다.**

```
from tensorflow.keras.losses import categorical_crossentropy
```

```
model.compile(optimizer = 'Adagrad',
              loss = 'categorical_crossentropy',
              metrics = ['acc'])
```

**Adagrade+Relu: 0.9159**

**5. Adam** : momentum의 장점 + Adagrade의 장점을 살린 경사 하강법

momentum의 장점 : 관성을 이용함

Adagrade의장점 : 러닝레이트가 자동 조절됨

**문제114. 3층 신경망의 경사하강법을 Adam으로 하고 학습 시키시오!**

**Adam + relu : 0.9737**

**6. Rmsprop** : Adagrade의 단점을 살려서 목표지점에 도달할 때 이전 내려오던 그 맥락( 걸음걸이 )을 살피서 러닝레이트를 조절한다.

**문제115. RMSprop으로 경사하강법을 변경해서 정확도를 보시오!**

SGD + relu : 0.9524

Adagrade+Relu: 0.9159

Adam + relu : 0.9737

RMSprop + relu : 0.9747

RMSprop 최고!

## ■ 가중치 초기값 설정 p202

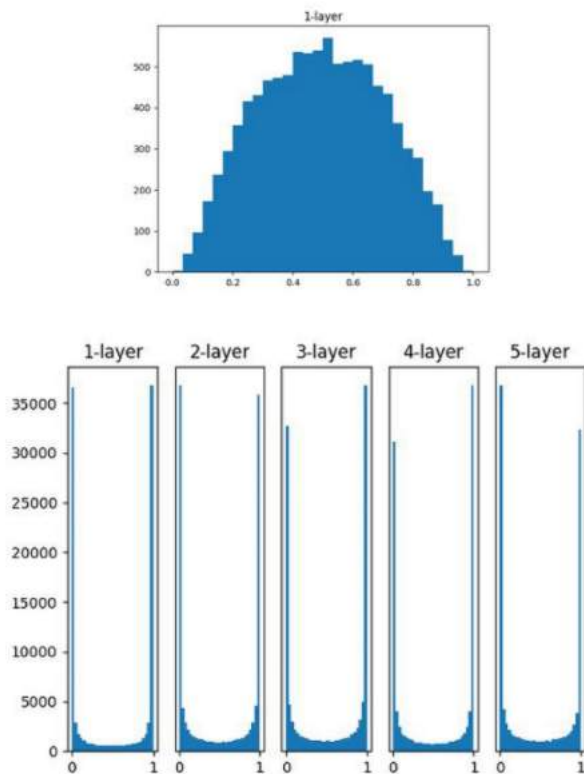
랜덤으로 생성되는 가중치  $w$ 의 초기값을 어떻게 선정하느냐에 따라 학습이 잘 될수도 있고 안 될수도 있다. 학습이 잘 되려면 가중치 초기값들의 분포가 정규분포 형태를 이루어야 한다. 그런데 `np.random.randn`은 가우시안 정규분포( 평균이 0이고 표준편차가 1)를 따르는 난수를 생성한다.

### 1. 생성되는 가중치의 표준편차가 너무 큰 경우

시험문제가 너무 어려우면 아주잘하는 학생들과 아주 못하는 학생들로 점수가 나뉜다.

구현예 : `1 * np.random.randn( 784, 50 )` #  $w_1$  생성

#### 가중치 초기화의 중요성

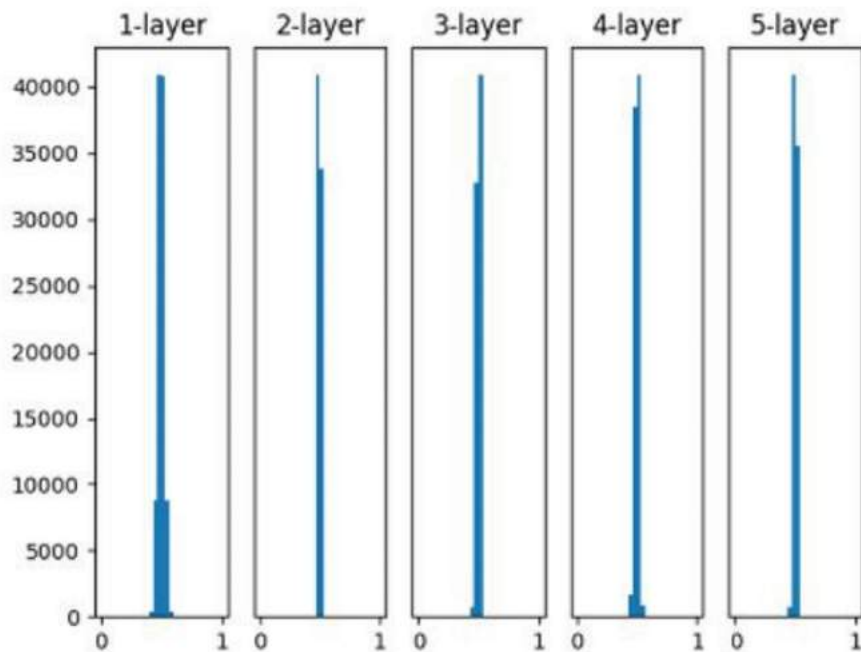


이렇게 되면 신경망 학습이 잘 안된다.

### 2. 생성되는 가중치의 표준편차가 너무 작은 경우

시험문제가 너무 쉬우면 학생들의 점수가 평균에 가까워진다.

구현예 : `0.01*np.random.randn( 784, 50 )`



### \* 다른 표준편차를 지정하는 방법 2가지

1. Xavier( 사비에르 ) 초기값 선정 ---> sigmoid 함수와 짝꿍
2. He 초기값 선정 ---> Relu 함수와 짝꿍 / 현업에서 많이 씀

#### # 1. 파이썬 날코딩 ( He 초기값 )

```
def __init__(self, input_size, hidden_size, output_size, weight_init_std=0.01):
    # 가중치 초기화
    self.params = {}
    self.params['W1'] = np.sqrt(2/input_size) * np.random.randn(input_size, hidden_size)
    self.params['b1'] = np.zeros(hidden_size)
    self.params['W2'] = np.sqrt(2/hidden_size) * np.random.randn(hidden_size, output_size)
    self.params['b2'] = np.zeros(output_size)
```

#### # 2. 텐서 2.0( He 초기값 )

# 5. 신경망 모델을 구성한다. ( 3층 신경망으로 구성 )

```
model = Sequential()
```

```
initializer = tf.keras.initializers.GlorotNormal()
```

```
model.add( Dense( 100, activation = 'relu', kernel_initializer = initializer,
input_shape=(784, ) ) ) #1층구성
```

```
model.add( Dense( 50, activation = 'relu' ) ) #2층구성
```

```
model.add( Dense( 10, activation='softmax' ) )
```

정확도 : 0.9757

## ■ underfitting을 방지하는방법 :

### ■ 배치 정규화 (batch normalization ) p 210

가중치 초기화 값을 적절히 설정하면 각 층의 활성화 값의 분포가 적당히 퍼지는 효과를 보이는데 신경망이 깊어지고 학습이 반복되다 보면 각 층의 활성화 값의 분포가 정규성을 잃어버리는 현상이 발생하게 된다.

그래서 이 정규성을 계속해서 유지 시키도록 '강제화'하는 방법이 배치 정규화이다.

# 5. 신경망 모델을 구성한다. ( 3층 신경망으로 구성 )

```
model = Sequential()
```

```
initializer = tf.keras.initializers.GlorotNormal()
```

```
model.add( Dense( 100, activation = 'relu', kernel_initializer = initializer,  
input_shape=(784, ) ) )#1층구성
```

```
model.add( BatchNormalization() )
```

```
model.add( Dense( 50, activation = 'relu' ) ) #2층구성
```

```
model.add( BatchNormalization() )
```

```
model.add( Dense( 10, activation='softmax' ) )
```

0.9765로 정확도 올라감

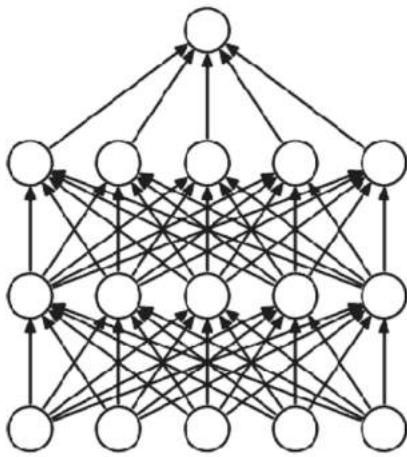
지금까지의 방법으로는 0.99를 넘길 수 없고 이미지 데이터이므로 7장에서 배울 CNN을 사용해야 0.99를 넘길 수 있다.

## ■ overfitting을 방지하는 방법:

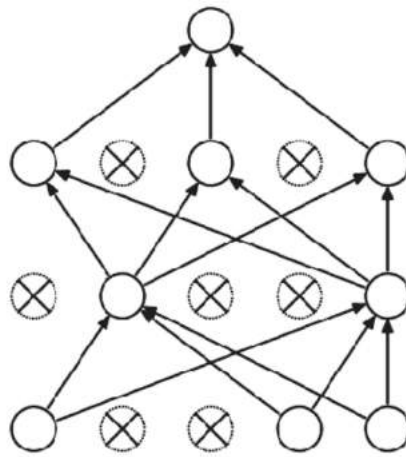
### ■ 드롭아웃( dropout ) p 219

"오버피팅을 억제하기 위해서 뉴런을 임의로 삭제하면서 학습시키는 방법"

그림6-22



(a) 일반 신경망



(b) 드롭아웃을 적용한 신경망

텐서 플로우 2.0

# 5. 신경망 모델을 구성한다. ( 3층 신경망으로 구성 )

```
model = Sequential()
```

```
initializer = tf.keras.initializers.GlorotNormal()
```

```
model.add( Dense( 100, activation = 'relu', kernel_initializer = initializer,
input_shape=(784, ) ) )#1층구성
```

```
model.add( BatchNormalization() )
```

```
model.add( Dropout(0.2) ) # 20퍼센트 제거
```

```
model.add( Dense( 50, activation = 'relu', kernel_initializer = initializer ) ) #2층구성
```

```
model.add( BatchNormalization() )
```

```
model.add( Dropout(0.2) )
```

```
model.add( Dense( 10, activation='softmax', kernel_initializer = initializer ) )
```

정확도 : 0.9791 정확도 더 올라감

## ■ 엘리스탑

1등 tensorflow + keras( tensorflow에서 인수 )

2등 pytorch

훈련 데이터의 정확도와 validation이 같아지는 부분에서 멈추겠다는 뜻

```
from tensorflow.python.keras.callbacks import EarlyStopping
```

```
early_stopping = EarlyStopping()

history = model.fit(x_train, y_train,
                    epochs = 100,
                    batch_size = 100,
                    validation_data = (x_val, y_val),
                    callbacks=[early_stopping])
```

## ■ L2정규화 ( 가중치 감소 ) p202

### 가중치 감소( Weight Decay )

" 학습하는 과정에서 큰 가중치에 대해서는 그에 상응하는 큰 패널티를 부여하여 오버피팅을 억제하는 방법 "

입력층 -----> 은닉층 -----> 출력층

입력층 -----> 은닉층 -----> 출력층

- -----> 귀가 있어요 -----> **아주 큰 가중치가 출력**
- -----> 꼬리가 있어요 -----> 가중치 출력
- -----> 집계발을 가지고 있어요 -----> 가중치 출력
- -----> 장난스럽게 보여요 -----> 가중치 출력

귀가 없는 고양이 사진이 들어오면 이 사진은 고양이가 아니다 라고 판단한다.

--> 오버피팅 일어남. 이때 해결하는 방법이 "가중치 감소" 이다.

입력층 -----> 은닉층 -----> 출력층

- -----> 귀가 있어요 -----> 아주 큰 가중치가 출력 <-- 큰 패널티
- -----> 꼬리가 있어요 -----> 가중치 출력
- -----> 집계발을 가지고 있어요 -----> 가중치 출력
- -----> 장난스럽게 보여요 -----> 가중치 출력

아주 큰 가중치에 대해서는 큰 패널티를 부여해서 가중치 매개변수의 값이 작아지도록 만드는 학습 방법이 가중치 감소이다.

# 5. 신경망 모델을 구성한다. ( 3층 신경망으로 구성 )

```
model = Sequential()
```

```
initializer = tf.keras.initializers.GlorotNormal()
```

```
model.add( Dense( 100, kernel_regularizer='l2', activation = 'relu',kernel_initializer = initializer,
```

```

input_shape=(784, ) ) #1층구성
model.add( BatchNormalization() )
model.add( Dropout(0.2) )

model.add( Dense( 50, activation = 'relu' ,kernel_initializer = initializer ) ) #2층구성
model.add( BatchNormalization() )
model.add( Dropout(0.2) )

model.add( Dense( 10, activation='softmax',kernel_initializer = initializer) )

```

## ■ 6장. 요약

### 1. 언더피팅을 막는 방법

1. 고급경사감소법
2. 가중치 초기값 설정
3. 배치 정규화

### 2. 오버피팅을 막는 방법

1. 드롭아웃
2. 엘리스탑
3. l2 정규화

## ■ 포트폴리오 준비 1

mnist 말고 다른사진을 신경망에 넣을 줄 알아야 한다.

스크롤링 한 사진이 있으면 그 사진을 쓰고 없으면 이파리 사진을 쓴다.

이파리 사진 : 훈련 데이터 19000장 ( 0~9500 : 정상 이파리,  
9501 ~ 19000 : 질병 이파리 )

테스트 데이터 1000장 ( 0~500: 정상 이파리,  
501~ 1000 : 질병 이파리 )

1. 신경망 코드( 이미 만들었고 계속해서 만들 것 )
2. 신경망에 사진을 로드하는 코드

```
test_image = 'D:\data\leafs\images\test'
```



```
import os

def image_load(path):
    file_list = os.listdir(path)
    return file_list

print( image_load(test_image) )
```

**예제2. 위의 결과에서 .jpg는 빼고 숫자만 출력되게 하시오!**

```
import os
import re

def image_load(path):
    file_list = os.listdir(path)
    file_name=[]
    for i in file_list:
        a = int( re.sub('[^0-9]', '', i) ) #i가숫자가아니면 null로 변경해라
        file_name.append(a)
    return file_name

print( image_load(leaf) )
```

**문제116. (오늘의 마지막 문제) 위의 결과가 정렬되서 출력되게 하시오**

**결과 : [ 1, 2, 3, 4, 5, ..... ]**

참고:

```
import os
import re

def image_load(path):
    file_list = os.listdir(path)
    file_name=[]
    for i in file_list:
        a = int( re.sub('[^0-9]', '', i) ) # i 가 숫자가 아니면 null 로 변경해라 ~
        file_name.append(a)
    return file_name

print ( image_load(test_image) )
```

답 :

```
import os
```

```
import re

def image_load(path):
    file_list = os.listdir(path)
    file_name=[]
    for i in file_list:
        a = int( re.sub('[^0-9]', '', i) ) #i가 숫자가 아니면 null로 변경
        file_name.append(a)
    return file_name

print( sorted(image_load(test_image)) )
```

03/17

2021년 3월 17일 수요일 오전 9:42

## ■ 딥러닝 cnn 총 복습

1장. numpy 사용법

2장. 퍼셉트론

3장. 3층 신경망 생성( 저자가 만들어온 가중치로 생성 )

4장. 2층 신경망 생성( 수치미분을 이용해서 학습을 직접 시켰다 )

5장. 3층 신경망 생성( 오차역전파를 이용해서 학습을 직접 시켰다 ) - tensorflow 2.0

6장. 언더피팅과 오버피팅을 줄이는 방법- tensorflow 2.0

7장. CNN

8장. 딥러닝의 역사

딥러닝 개발자( 연구원 ), AI 개발자, 머신러닝 데이터 분석가

딥러닝 포트폴리오, 텐서플로우, 파이토치 사용가능하다는 걸 보여주기

개인 포트폴리오( 신경망 활용 홈페이지 생성 )

## ■ 6장. 언더피팅과 오버피팅을 줄이는 방법 - tensorflow 2.0

### 1. 언더피팅을 줄이는 방법

- 고급 경사감소법 종류( SGD, momentum, Adagrade, Adam )
- 가중치 초기값 설정
- 배치 정규화 ( 현업에서 아주 중요하게 사용되는 부분 ) ->  
층이 깊어져도 가중치가 계속 정규분포를 유지할 수 있도록 잡아주는 역할을 함  
오버피팅, 언더피팅 둘 다 줄여줌

### 2. 오버피팅을 줄이는 방법

- 드롭아웃
- 가중치 감소( L2, L4 )
- 엘리스탑 : 훈련데이터와 validation 데이터의 정확도가 같을 때 멈춤

이파리 문제

**예제3. 출력되는 숫자를 정렬해서 결과를 출력하시오.**

```
import os  
import re
```

```
def image_load(path):
    file_list = os.listdir(path)
    file_name=[]
    for i in file_list:
        a = int( re.sub('[^0-9]', '', i) ) #i가숫자가아니면 null로 변경해라
        file_name.append(a)
    file_name.sort()
    return file_name

print( image_load(test_image) )
```

**예제4. 위에서 출력된 결과를 아래와 같이 .jpg가 붙어서 출력되게 하시오.**

[1.jpg, 2.jpg, 3.jpg, 4.jpg, 5.jpg ... ]

```
import os
import re

def image_load(path):
    file_list = os.listdir(path)
    file_name=[]
    for i in file_list:
        a = int( re.sub('[^0-9]', '', i) ) #i가숫자가아니면 null로 변경해라
        file_name.append(a)
    file_name.sort()
    file_res = []
    for j in file_name:
        file_res.append('%d.jpg'%j)
    return file_res

print( image_load(test_image) )
```

**예제5. 이미지 이름 앞에 절대경로가 붙게 하시오!**

( 'D:\ww\data\ww\leafs\ww\images\ww\test1.jpg', 'D:\ww\data\ww\leafs\ww\images\ww\test2.jpg', ... )

```
import os
import re

def image_load(path):
    file_list = os.listdir(path)
    file_name=[]
    for i in file_list:
        a = int( re.sub('[^0-9]', '', i) ) #i가숫자가아니면 null로 변경해라
        file_name.append(a)
    file_name.sort()
    file_res = []
    for j in file_name:
```

---

```

        file_res.append('%s%d.jpg'%(path,j))
    return file_res

print( image_load(test_image) )

```

## 예제6. opencv를 이용해서 이미지를 numpy array 형태의 숫자로 변환하시오!

```

import os
import re
import cv2
import numpy as np

def image_load(path):
    file_list = os.listdir(path)
    file_name=[]
    for i in file_list:
        a = int( re.sub('[^0-9]', '', i) ) #i가숫자가아니면 null로 변경해라
        file_name.append(a)
    file_name.sort()
    file_res = []
    for j in file_name:
        file_res.append('%s%d.jpg'%(path,j))
    return file_res

    image = []
    for k in file_res:
        img = cv2.imread(k)
        image.append(img)

    return np.array(image)

print( image_load(test_image) )

```

※ 신경망에 이미지를 넣으려면 2개의 함수가 있어야 한다.

1. image\_load : 이미지를 숫자로 변환해주는 함수
2. label\_load : 정답 라벨을 onehot encoding 해주는 함수

**이파리 사진 : 훈련 데이터 19000장** ( 1~9500 : 정상 이파리,  
9501 ~ 19000 : 질병 이파리 )

테스트 데이터 1000장 ( 1~500: 정상 이파리,  
501~ 1000 : 질병 이파리 )

## 예제7. 훈련데이터의 라벨을 train\_label.csv로 생성하시오!

( 1~ 9500을 1로 하고 9501-19000을 0으로 해서 만드시오 )

```
path="D:\\data\\leafs\\images\\train_label.csv"
```

```
file = open( path, 'w')
```

```
for i in range( 0, 9500):
```

```
file.write( str(1) + '\n' )
```

```
for i in range( 0, 9500 ):
```

```
file.write( str(0) + '\n' )
```

```
file.close()
```

### 예제8. 테스트 데이터의 라벨을 test\_label.csv로 생성하세요!

테스트 데이터 1000장 ( 1~500: 정상 이파리, 501~ 1000 : 질병 이파리 )

```
path="D:\\data\\leafs\\images\\test_label.csv"
```

```
file = open( path, 'w')
```

```
for i in range( 0, 500):
```

```
file.write( str(1) + '\n' )
```

```
for i in range( 0, 500 ):
```

```
file.write( str(0) + '\n' )
```

```
file.close()
```

### 예제9. train label.csv의 내용을 불러오는 함수를 생성하시오!

```
train_label = "D:\\data\\leafsw\\images\\train_label.csv"
```

```
import csv
```

```
def label_load(path):
```

```
file = open(path)
```

```
labeldata = csv.reader(file)
```

```
labellist = []
```

```
for i in labeledata:
```

```
labellist.append(i)
```

```

return labellist

```

```
print( label_load( train_label ) )
```

[illegible]

[ '1' ],  
[ '1' ], [ '1' ], [ '1' ], [ '1' ], [ '1' ], [ '1' ], [ '1' ], [ '1' ], [ '1' ] ... ]

**예제 10. 위의 결과는 문자로 출력이 되었는데 숫자로 출력되게 하시오!**

```
train_label = "D:\\data\\leafs\\images\\train_label.csv"
import csv

def label_load(path):
    file = open(path)
    labeledata = csv.reader(file)
    labellist = []
    for i in labeledata:
        labellist.append(i)
    label = np.array(labellist) # 리스트를 numpy array로 변환한다.
    label = label.astype(int) #숫자변환
    return label

print( label_load( train_label ) )
```

**예제11. 위의 숫자를 one hot encoding 하기 위해 아래의 코드를 연습하시오!**

```
import numpy as np

print( np.eye(10)[4] ) # 숫자 4가 one hot encoding 된다.
```

## 예제12. label\_load 함수의 결과가 one hot encoding된 결과로 출력되게 하시오!

```
train_label = "D:\ww\data\ww\leafs\ww\images\ww\train_label.csv"
import csv

def label_load(path):
    file = open(path)
    labeledata = csv.reader(file)
    labellist = []
    for i in labeledata:
        labellist.append(i)

    label = np.array(labellist) # 리스트를 numpy array로 변환한다.
    label = label.astype(int) #숫자변환
    label = np.eye(2)[label]
    return label

print( label_load( train_label ) )
```

**예제13. 위의 결과는 3차원인데 신경망에서 라벨로 사용하려면 2차원이어야 하므로 차원을 2차원으로 축소시켜서 출력하시오!**

```
train_label = "D:\\data\\leafs\\images\\train_label.csv"
import csv

def label_load(path):
    file = open(path)
    labeldata = csv.reader(file)
    labellist = []
    for i in labeldata:
        labellist.append(i)
    label = np.array(labellist) # 리스트를 numpy array로 변환한다.
    label = label.astype(int) #숫자변환
    label = np.eye(2)[label] # one hot encoding 한다.
    label = label.reshape(-1,2) # -1은 이 자리에 뭐가 올지 몰라서 -1을 쓴것이고 2는 one hot
encoding이 [0,1]이렇게
                                # 생겨서 이렇게 써준것이다.

    return label

print( label_load( train_label ).shape )
```

**예제14. 위에서 만든 함수 2개를 loader\_leaf.py로 저장하시오!**

```
import numpy as np
import cv2
import csv
import os
import re

def image_load(path):
    file_list = os.listdir(path)
    file_name=[]
    for i in file_list:
        a = int( re.sub('[^0-9]','', i) ) # i 가 숫자가 아니면 null 로 변경해라 ~
        file_name.append(a)
    file_name.sort() # 정렬작업

    file_res=[]
    for j in file_name:
        file_res.append('%s\\%d.jpg' %(path,j) )

    image=[]
    for k in file_res:
```



```

        img = cv2.imread(k)
        image.append(img)

    return np.array(image)

def label_load(path):
    file = open(path)
    labeldata = csv.reader(file)
    labellist = []
    for i in labeldata:
        labellist.append(i)
    label = np.array(labellist) # 리스트를 numpy array 로 변환한다.
    label = label.astype(int) # 숫자변환
    label = np.eye(2)[label] # one hot encoding 한다.
    label = label.reshape(-1,2)
    return label

```

## 예제15. 어제 만들었던 mnist 용 3층 신경망에 이파리 데이터를 로드하고 학습시키시오!

```

import loader_leaf

test_image = "D:\data\leafs\images\test"
test_label = "D:\data\leafs\images\test_label.csv"

print(loader_leaf.image_load(test_image))

```

## 예제15. 이파리 사진을 256x256에서 32x32로 resize 하시오!

```

for k in file_list: # 리스트 안에 있는 파일들을 하나씩 빼내는 코드
    img = cv2.imread(path + '\ ' + k) # 설현 사진을 숫자행렬로 변경한다.
    print( img )

for k in file_list: # 리스트 안에 있는 파일들을 하나씩 빼내는 코드
    img = cv2.imread(path + '\ ' + k) # 설현 사진을 숫자행렬로 변경한다.
    width, height = img.shape[:2] # 설현사진 숫자 행렬에서 가로, 세로를 가져온다.
    resize_img = cv2.resize(img, (32 , 32), interpolation=cv2.INTER_CUBIC)
    cv2.imwrite('d:\data\leafs\images\train' + k, resize_img) # resize한 이미지를 저장한다.

plt.imshow(resize_img) # resize 된 사진을 시각화
plt.show()

```

## 예제16. ( 점심시간 문제 ) 이파리 사진 테스트 이미지를 256x256에서 32x32로 resize 하시오!

```
import cv2
import os
import numpy as np

path = "D:\\data\\leaves\\images\\test"
file_list = os.listdir(path) # path 에 지정된 위치에 있는 파일들의 이름을 불러온다.
file_list = ['b.jpg']

import numpy as np

for k in file_list: # 리스트 안에 있는 파일들을 하나씩 빼내는 코드
    img = cv2.imread(path + '\\'+ k) # 수지 사진을 숫자행렬로 변경합니다.
    #
    width, height = img.shape[:2] # 수지 사진 숫자 행렬에서 가로, 세로 가져온다.
    resize_img = cv2.resize(img, (32, 32), interpolation=cv2.INTER_CUBIC)
    cv2.imwrite('D:\\data\\leaves\\images\\test_resize2\\'+ k, resize_img) # resize 한 이
    미지를 저장합니다.
```

## ■ 7장. CNN( Convolution Neural Network )

Convolution : 합성곱

Neural : 신경

Network : 망

### 합성곱 신경망 ?

convolution 층과 pooling 층을 포함하는 신경망

### 기존 신경망과의 차이점?

- 기존방법 : 입력값 ---> Affine 계층 ---> Relu ( 완전 연결 계층 )

↑

사진을 사진 그대로 입력하지 않고 1차원으로 flatten 시켜서 입력함

- CNN : 입력값 ---> Convolution ---> Relu ---> pooling ---> 완전연결계층

↑

사진을 사진 그대로 입력

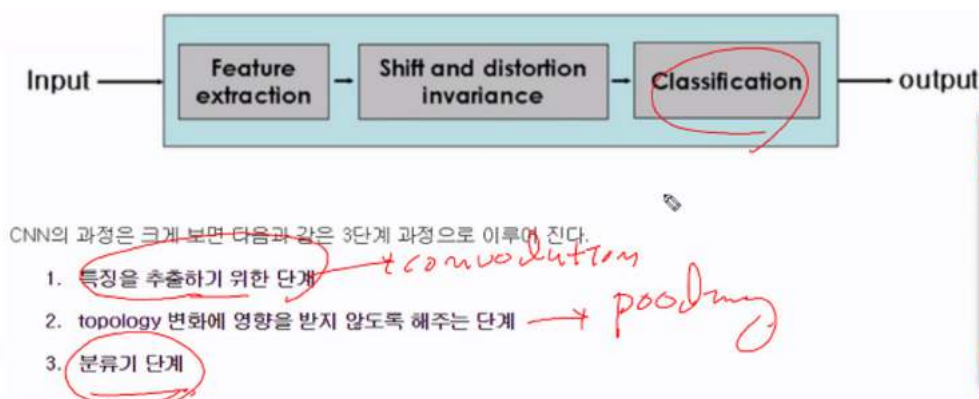
Convolution 층 : 이미지의 특징을 잡아낸 어떤 이미지 생성

pooling 층 : 이미지를 선명하게 만드는 역할

## ■ cnn을 이용하지 않았을 때의 기존층의 문제점 ?

<https://cafe.daum.net/oracleoracle/Sedp/199>

" 이미지의 형상이 무시된다. "



분류기 단계 : classification 단계

정리 : 필기체 이미지 --->  $28 \times 28 = 784$ 의 1차원 데이터로 변경해서 784개의 데이터를 척 affine 계층에 입력한게 **기존 방법**이다.

기존방법의 문제점 ?

형상을 무시하고 모든 입력 데이터를 동등한 뉴런으로 취급하기 때문에 이미지가 갖는 본질적인 패턴을 읽지 못한다.

문제점을 해결하는 방법 ?

원본 이미지를 가지고 여러개의 feature map을 만들어서 데이터를 여러 개 생성한다.

( cnn층에서 이 일이 일어난다. )

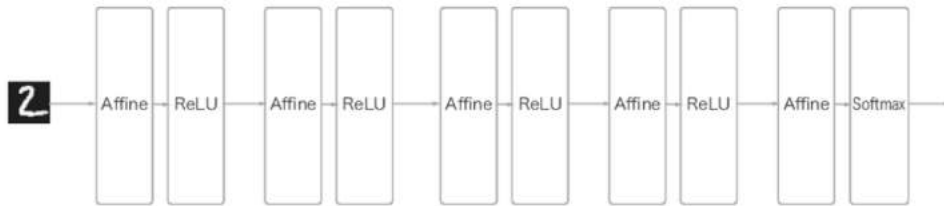
## ■ 합성곱 연산을 컴퓨터로 구현하는 방법 ?

원본 이미지 한장을 filter 100개로 합성곱 연산을 하면 feature map이 100개가 생성된다.

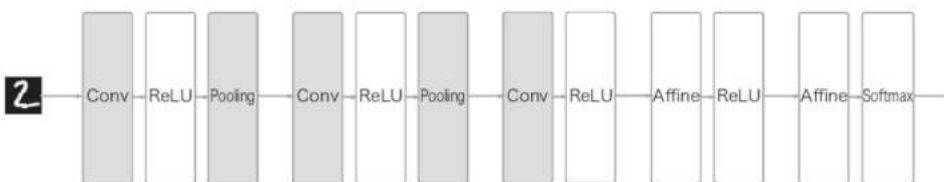
## 합성곱의 역할?

이미지의 특징을 잡아내는 역할을 한다.

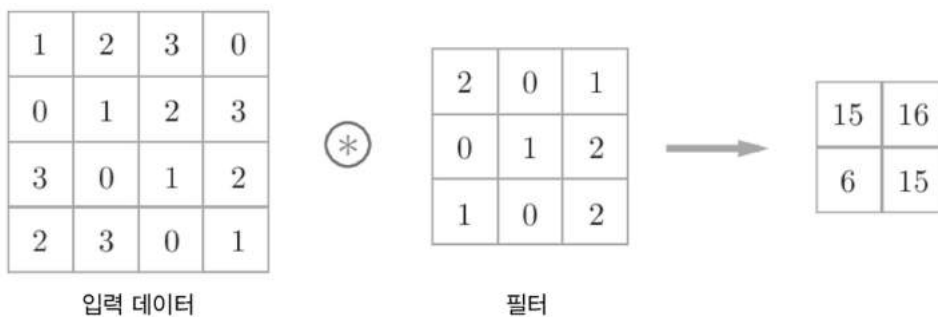
### 1. 완전연결계층만 있었을 때



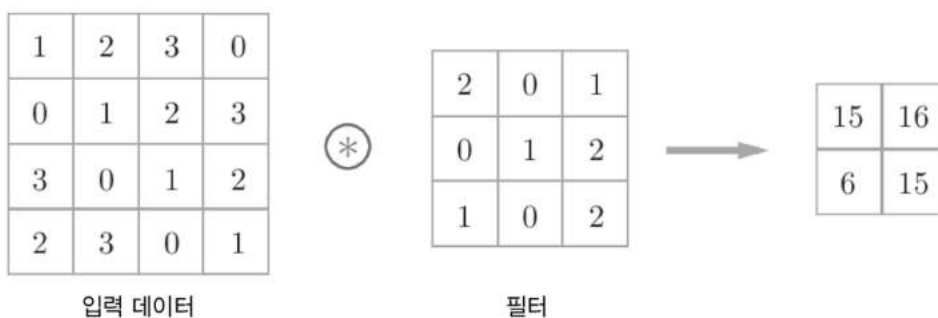
### 2. cnn + 완전연결계층



### 3. 합성곱 연산



문제117. 합성곱을 하기 위해서 입력 데이터와 filter를 만드시오!



```
import numpy as np
```

```
x = np.array([[1,2,3,0],[0,1,2,3],[3,0,1,2],[2,3,0,1]])  
print(a)
```

```
filter = np.array( [2,0,1,0,1,2,1,0,2] ).reshape(3,3)
print(x)
print(filter)
```

**문제118. 입력 이미지 x에서 아래의 3x3 영역만 가져오시오!**

```
print(x[0:3,0:3])
```

**결과 :**

```
[[1 2 3]
 [0 1 2]
 [3 0 1]]
```

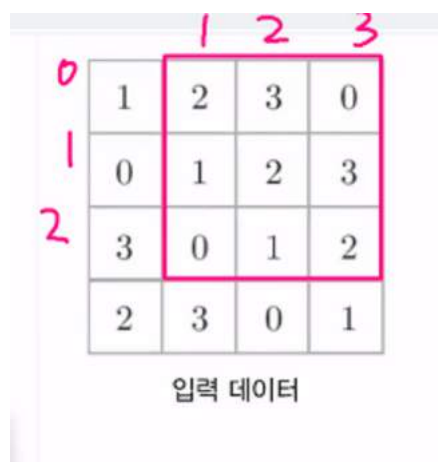
**문제119. 위에서 가져온 3x3 영역의 원소들과 filter의 원소들과 곱셈을 하시오!**

```
print(x[0:3,0:3] * filter )
```

**문제120. 위의 곱셈을 한 결과의 원소들을 다 더한 값은 무엇인가?**

```
print(np.sum(x[0:3,0:3] * filter) ) # 15
```

**문제121. 원본 이미지에서 아래의 영역과 filter를 곱한 원소들의 합은 얼마인가?**



		1	2	3
0	1	2	3	0
1	0	1	2	3
2	3	0	1	2
	2	3	0	1

입력 데이터

```
print(np.sum(x[0:3,1:4] * filter ))
```

**문제122. 결과적으로 나와야하는 4개의 숫자를 다 출력하시오!**

```
print(np.sum(x[0:3,0:3] * filter) )
print(np.sum(x[0:3,1:4] * filter ))
print(np.sum(x[1:4,0:3] * filter) )
print(np.sum(x[1:4,1:4] * filter) )
```

**문제123. 위의 4개의 계산을 위와 같이 4번 하드코딩하지 않고 이중 루프문으로 수행하시오!**

```
for i in range(2):
    for j in range(2):
        print( np.sum(x[ i:3+i,j:3+j ] * filter) )
```

**문제124. 위의 숫자 4개를 a라는 비어있는 리스트에 append 시키시오!**

```
a = []
for i in range(2):
    for j in range(2):
        a.append( np.sum(x[ i:3+i,j:3+j ] * filter) )
```

**문제125. 위의 a리스트의 shape를 2x2로 변경하시오!**

```
a = []
for i in range(2):
    for j in range(2):
        a.append( np.sum(x[ i:3+i,j:3+j ] * filter) )

b = np.array(a).reshape(2,2)
print(b)
```

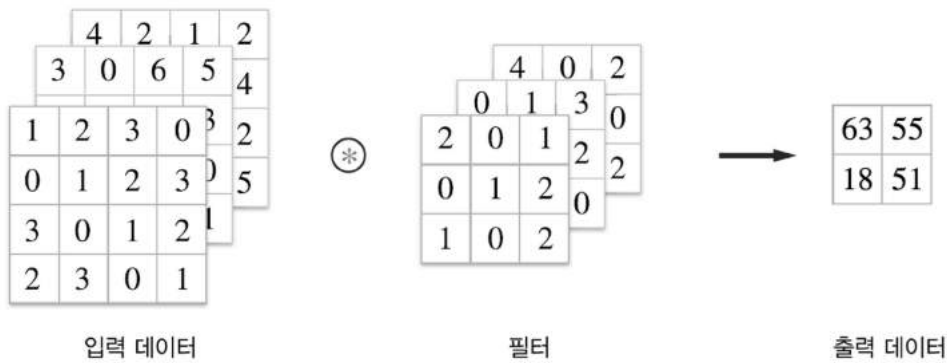
**문제126. 위의 결과에서 아래 그림의 편향 3을 더하시오!**

```
a = []
for i in range(2):
    for j in range(2):
        a.append( np.sum(x[ i:3+i,j:3+j ] * filter) )

b = np.array(a).reshape(2,2)
print(b + 3)
```

**지금까지 한 것은 그냥 합성곱이고 진짜로 해야하는 것은 3차원 합성곱이다.**

## ■ 3차원 합성곱



```
import numpy as np
```

```
x=np.array([[[[1,2,3,0],
               [0,1,2,3], # --> red 행렬
               [3,0,1,2],
               [2,3,0,1]],
             [[2,3,4,1],
               [1,2,3,4], # --> green 행렬
               [4,1,2,3],
               [3,4,1,2]],
             [[3,4,5,2], # --> blue 행렬
               [2,3,4,5],
               [5,2,3,4],
               [4,5,2,3]]]])
```

```
filter=np.array([[[[2,0,1],
                   [0,1,2],
                   [1,0,2]],
                  [[3,1,2],
                   [1,2,3],
                   [2,1,3]],
                  [[4,2,3],
                   [2,3,4],
                   [3,2,4]]]])
```

```
print(x)
print(filter)
```

**문제127. 3차원 합성곱을 하기 전에 먼저 x2 입력 이미지에서 red 행렬만 출력하십시오!**

```
import numpy as np

x2=np.array([[[[1,2,3,0],
               [0,1,2,3], # --> red 행렬
               [3,0,1,2],
               [2,3,0,1]],
              [[2,3,4,1],
               [1,2,3,4], # --> green 행렬
               [4,1,2,3],
               [3,4,1,2]],
              [[3,4,5,2], # --> blue 행렬
               [2,3,4,5],
               [5,2,3,4],
               [4,5,2,3]]]])
```

```
print( x2.shape )
print( x2[ 0 , : , : ] )
```

## 문제128. 3차원 합성곱을 진행하시오!

```
a = []

for k in range(3): # 0,1,2 색상별로 합성곱을 진행하기 위한 for문
    for i in range(2):
        for j in range(2):
            a.append( np.sum(x[k, i:3+i,j:3+j ] * filter[k, :, :]) ) # 색상, 가로, 세로

print(a) #[15, 16, 6, 15, 46, 48, 36, 46, 95, 98, 84, 95]

b =np.array(a).reshape( 3,4 ) #3행 4열로 변경
print(b)

print(np.sum(b, axis=0)) #axis =0이면 세로로 더한다.
                        # axis = 1이면 가로로 더한다.

c = np.sum(b,axis=0)
print(c.reshape(2,2))
```

**결과 :**

**feature map:**

**[[156 162]**

**[126 156]]**

※ 3차원 원본 이미지 x에 3차원 filter를 대고 2차원 feature map을 생성함



**문제129. 유럽.png를 아래의 3차원 필터로 합성곱하기 위하여 유럽.png를 numpy array 형태의 숫자로 변환하시오!**

```
from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

img = Image.open('유럽.png')
img_pixel = np.array(img)
print(img_pixel.shape) # (499, 756, 3)
```

**문제130. 유럽 이미지랑 합성곱 할 필터를 아래와 같이 랜덤으로 생성하시오!**

```
filter3 = np.random.rand( 5,5,3 )
print(filter3)
print(filter3. shape) #(5, 5, 3)
```

**문제131. 문제 128번에서 사용했던 3차원 합성곱 코드를 가져와서 3차원 유럽 사진에 3차원 필터와 합성곱해서 feature map을 생성하시오!**

```
print( img_pixel.shape ) # (499, 756, 3)
print(filter3.shape ) # (5, 5, 3)

row = 499-5 + 1
col = 756-5 + 1

a = []

# 0, 1, 2 색상별로 합성곱을 진행하기 위해서
for i in range(row):
    for j in range(col):
        b=0
        for k in range(3):
            b += np.sum(img_pixel [i: 5 + i, j:5+j, k] *filter3[:, :, k] )
        a.append(b)

c = np.array(a).reshape(495,752)
```

**문제132. ( 오늘의 마지막 문제 ) 유럽사진의 피쳐맵 시각화된 것을 올리세요!**

```

from PIL import Image
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

img = Image.open('유럽.png')
img2 = Image.open('유럽2.png')
img_pixel = np.array(img)
img_pixel2 = np.array(img2)
print(img_pixel.shape) # (499, 756, 3)
print(img_pixel2.shape)

filter3 = np.random.rand( 5,5,3 )

print(filter3. shape) #(5, 5, 3)
plt.imshow(filter3)
plt.show()

print( img_pixel.shape ) # (499, 756, 3)
print(img_pixel2.shape)
print(filter3.shape ) # (5, 5, 3)

row = 499-5 + 1
col = 756-5 + 1

row2 = 416-5+1
col2 = 705-5+1

a = []
x = []

# 0, 1, 2 색상별로 합성곱을 진행하기 위해서
for i in range(row):
    for j in range(col):
        b=0
        for k in range(3):
            b += np.sum(img_pixel [i: 5 + i, j:5+j, k] *filter3[:, :, k] )
        a.append(b)

for i in range(row2):
    for j in range(col2):
        y=0
        for k in range(3):
            y += np.sum(img_pixel2[i: 5 + i, j:5+j, k] *filter3[:, :, k] )
        x.append(y)

c = np.array(a).reshape(495,752)
z = np.array(x).reshape(412, 701)

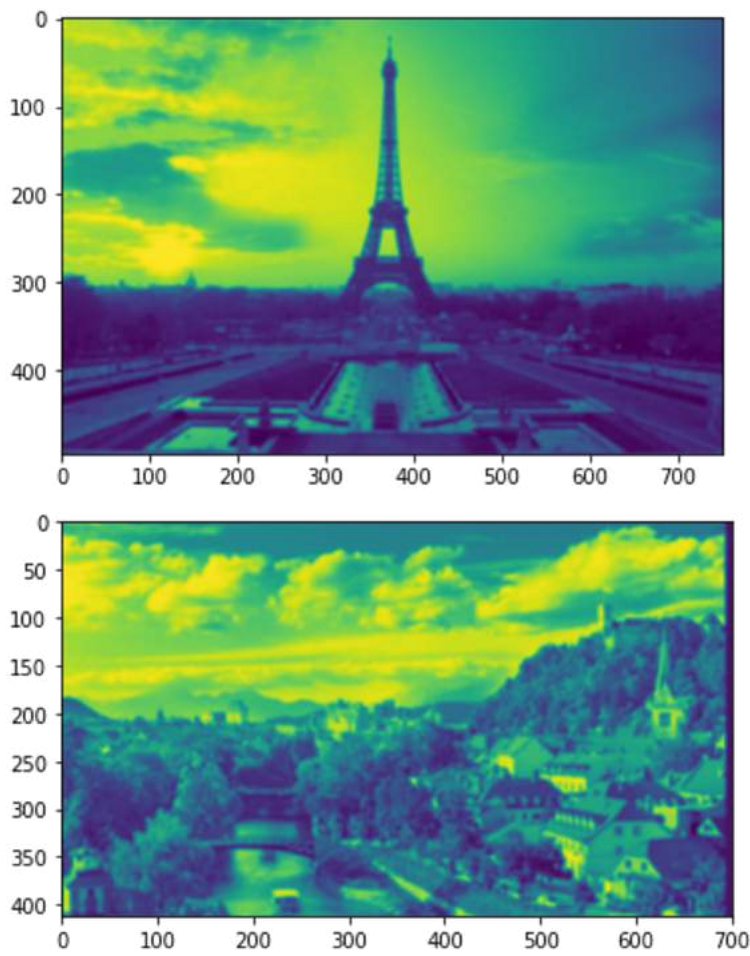
plt.imshow(c) # resize 한 수지 사진을 시각화 해라 ~

```

```
plt.show()
```

```
plt.imshow(z)
```

```
plt.show()
```



## ■ 7장. cnn 복습

1. 기존 완전연결 계층의 문제점
2. 합성곱 연산 : 이미지에서 특징을 추출하는 feature map 생성
3. 3차원 합성곱 연산 : 칼라 이미지에서 특징을 추출하는 feature map 생성

- 이론을 이해 : conv 함수 --> 공부한 것
- 코드 구현 이해 ( for 문 3개 사용 ) ---> 결과 산출물( 일했다 )

입력 데이터 -----> 합성곱 층 -----> 완전연결 계층 -----> 분류결과  
(이미지)

**합성곱 층** : 이미지 형상을 유지, 한장의 사진을 가지고 여러개의 비슷한 사진들( FEATURE MAP )을 만들어 냄

**완전연결 계층** : 분류를 하는 계층

### 4. 패딩

패딩이 무엇인데 합성곱층에서 패딩이 필요한가?

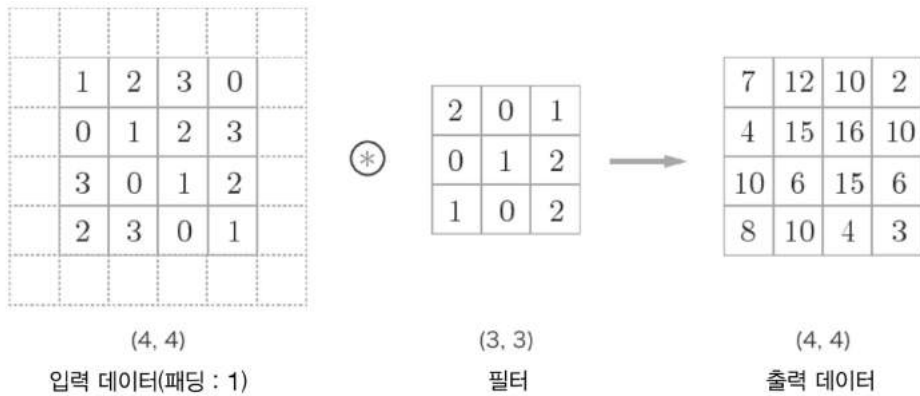
패딩이란 합성곱 연산을 수행하기 전에 입력 데이터 주변을 특정 값으로 채워 늘리는 것을 패딩이라고 한다.

**합성곱 과정을 다시 살펴 보면? ( 그림 7-3 )**

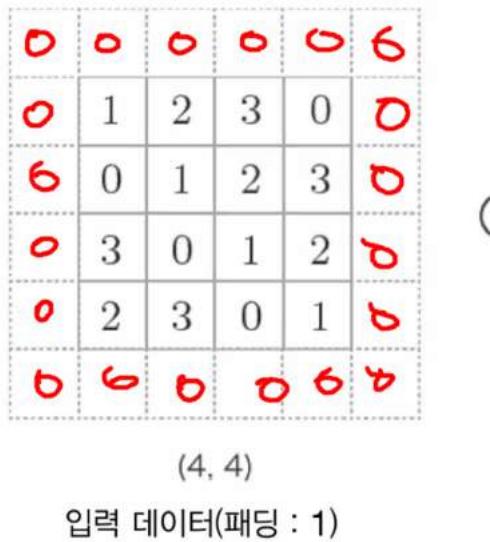


패딩을 하지 않을 경우 data의 공간 크기는 합성곱 계층을 지날때마다 작아지게 되므로 가장자리 정보들이 사라지게 되는 문제가 발생한다. 그래서 패딩을 해야한다.

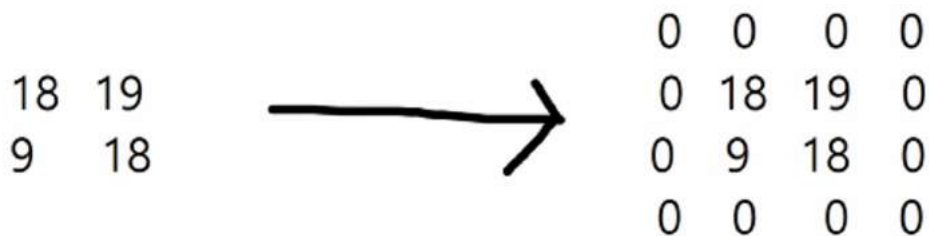
패딩은 다음 그림 7-6과 같다.



*4x4 → 6x6*



예제1. 아래의 2x2 행렬을 제로 패딩1 해서 4x4 행렬로 만드시오!



```
import numpy as np
```

```
x = np.array( [[ 18, 19 ], [ 9, 18 ] ] )
```

```
x_pad = np.pad( x, pad_width=1, mode='constant', constant_values = 0 )
```

```
print( x_pad )
```

# pad\_width = 양쪽으로 하나씩만 채워넣기

```
# mode = 'constant' : 상수 채워넣기
# constant_values = 넣을 값 : 0
```

**문제133. 아래의 행렬을 제로패딩 1 한 후에 결과를 확인하시오!**

```
2 0 1
6 7 3
4 5 2
```

```
import numpy as np
```

```
x2 = np.array( [ [2,0,1], [6,7,3], [4,5,2]])
x_pad2 = np.pad( x2, pad_width =1, mode = 'constant', constant_values = 0)
print(x_pad2)
```

또는

```
x2( [ 2,0,1,6,7,3,4,5,2] ).reshape( 3,3 )
x2_pad = np.pad( x2, pad_width = 1, mode = 'constant', constant_values = 0)
print( x2_pad )
```

**결과 :**

```
[[0 0 0 0 0]
 [0 2 0 1 0]
 [0 6 7 3 0]
 [0 4 5 2 0]
 [0 0 0 0 0]]
```

**※ 패딩을 사용해서 우리가 하고자 하는 것은?**

- 원본이미지를 convolution 층을 지날때 마다 계속 같은 형상으로 유지 시키는 것

원본 입력이미지 ---> 컨볼루션층 ---> 원본 이미지와 행과 열이 같은 이미지

컨볼루션층 : convolution + padding 같이 함

원본 입력이미지와 같은 형상( 행과 열 )으로 피쳐맵이 출력되기 위해서는 패딩을 몇을 줘야하는가?

파이썬 날코딩 : padding = ? ( 개발자가 직접 계산해줘야 한다 )

텐서플로우 코드 : padding = same

계산 공식 :

계산공식 :

$$P = \frac{(OH - 1) * S - H + FH}{2} = \frac{(4 - 1) * 1 - 4 + 3}{2} = 1$$

$$OH = \frac{H + 2P - FH}{S} + 1$$

$$OW = \frac{W + 2P - FW}{S} + 1$$

OH : 출력 이미지의 세로

OW : 출력 이미지의 가로

H : 입력 이미지의 세로

W : 입력 이미지의 가로

FH : 필터의 세로

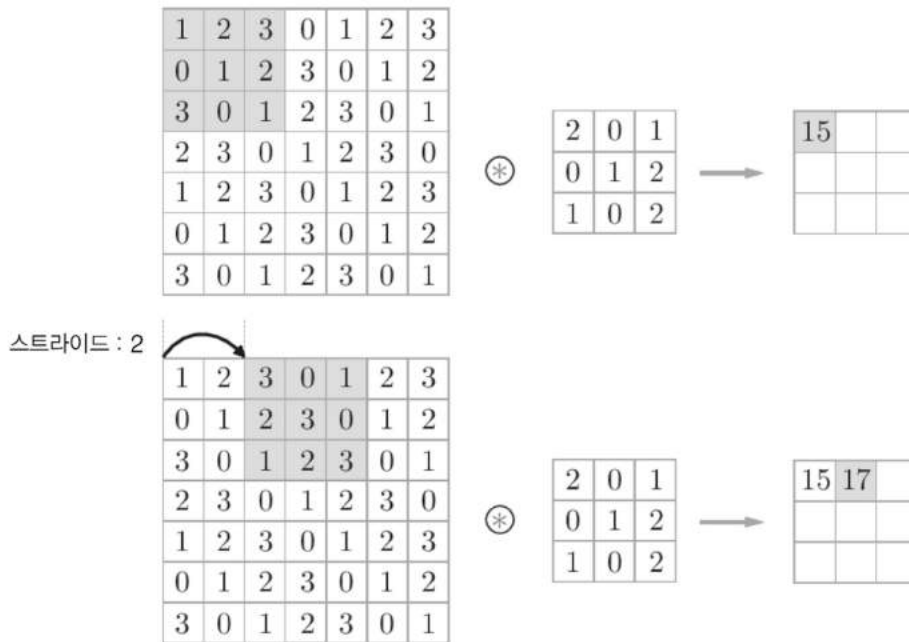
FW : 필터의 가로

**책 251페이지의 코드**

```
conv_param = ( 'filter_num': 30, 'filter_size' : 5, 'pad':0, 'stride':1 )
```

## ■ 스트라이드

합성곱할 때 원본 이미지를 필터로 이동하면서 피쳐맵을 생성하는 과정  
이동을 몇 칸으로 할 지를 결정하는 것



## 1. 텐서플로우 2.0으로 구현

```
model.add(Conv2D(32, (3, 3), padding='same', input_shape=x_train.shape[1:]))
```

# 32 : 32개의 층으로 하겠다.

# ( 3, 3 ) : 필터 사이즈( 가로, 세로 )

# padding = 'same' : 입력이미지와 출력이미지의 사이즈를 같게 하겠다.

# input\_shape : 입력층의 뉴런의 개수

## 2. 파이썬 날코딩( p 251 )

```
conv_params = ( 'filter_num' : 32, 'filter_size' : 3, 'pad' : 1, 'stride': 1 )
```

#'pad' : 계산된 값을 넣어줌, 근데 보통 1을 넣어준다.

# 'stride' : default =1

※ 컨볼루션 층에서 하는 작업은 원본 이미지를 가지고 여러개의 비슷한 이미지를 만들어 내는 작업이다. 그 비슷한 이미지들이 바로 feature map이다. 그리고 이 feature map의 개수는 필터의 개수와 동일하게 생성된다.

**아래와 같이 convolution 층을 구성했으면 생성되는 feature map의 개수는?**

```
model.add(Conv2D(32, (3, 3), padding='same', input_shape=x_train.shape[1:]))
```

답 : 필터의 개수( = 뉴런의 개수 )가 32개이므로 이미지 한장에 대해서 피쳐맵의 개수도 32개가 된다.



그런데 신경망에 이미지를 학습시킬 때 한장씩 시키지 않는다. 배치단위로 100장씩 한번에 넣어서 학습시킨다.

100장 ----> convolution -----> 피쳐맵의 개수 ? 3200장

↓

```
model.add(Conv2D(32, (3, 3), padding='same', input_shape=x_train.shape[1:]))
```

신경망이 잘 머리속에 이해되게 하려면 그림으로 이해하는게 좋은데 그 방법이 **블록으로 생각하기** 이다.

## ■ 블록으로 생각하기( p 237 )

3차원 합성곱 연산은 데이터와 필터를 직육면체 블록으로 생각하면 쉽다.  
블록은 아래의 그림과 같은 3차원 직육면체 이다.

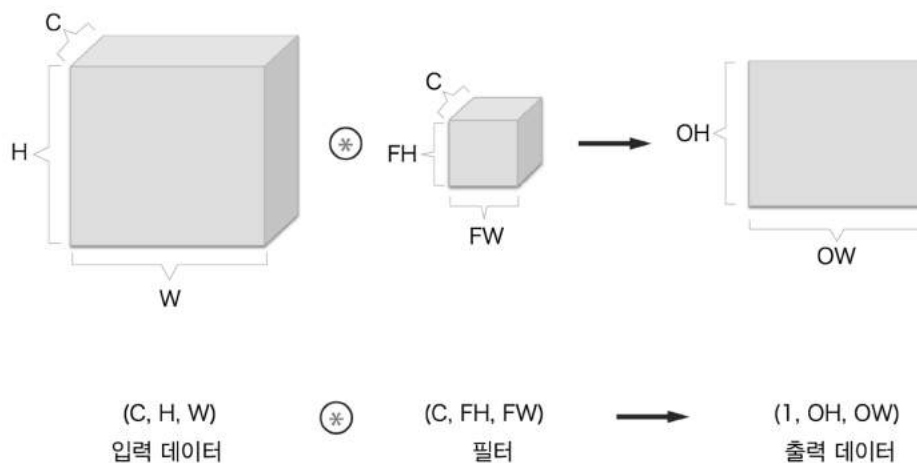
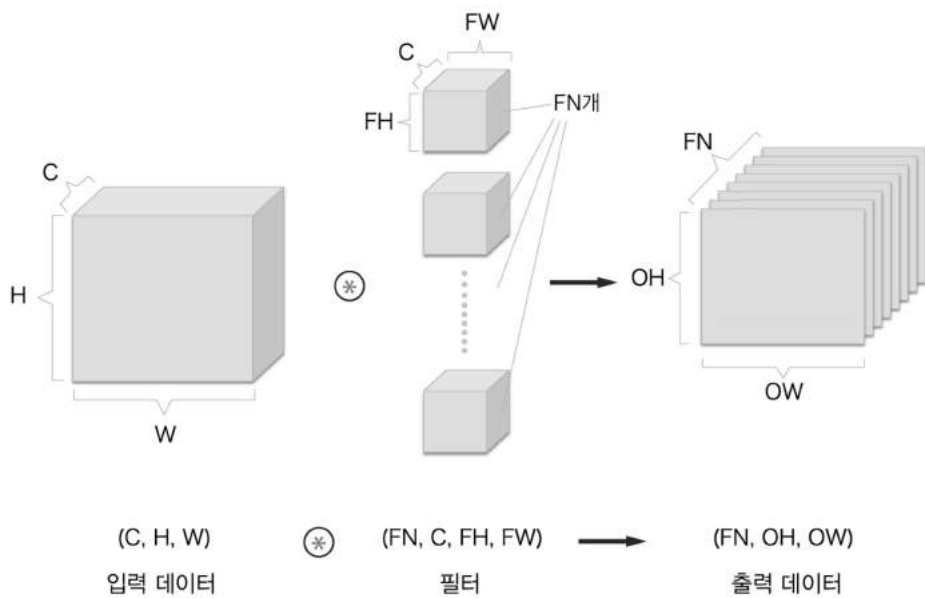


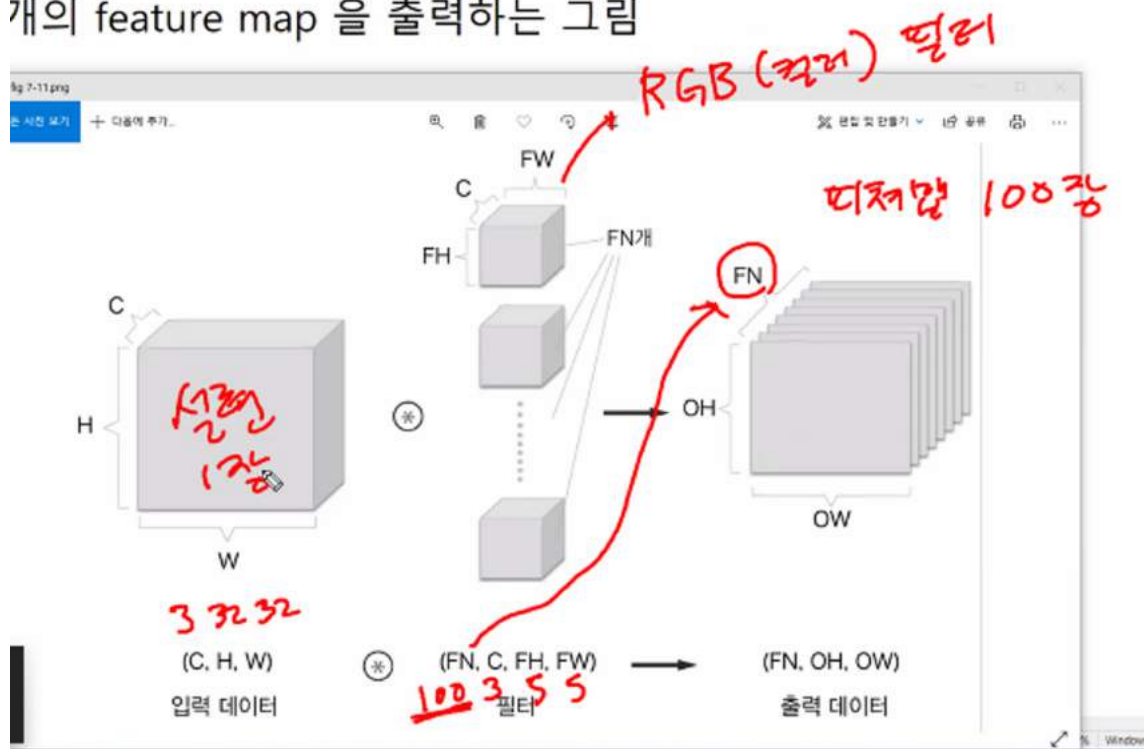
그림 7-10

설현 사진 RGB 컬러 사진 1장을 RGB 필터로 합성곱해서 1개의 feature map을 출력하는 그림



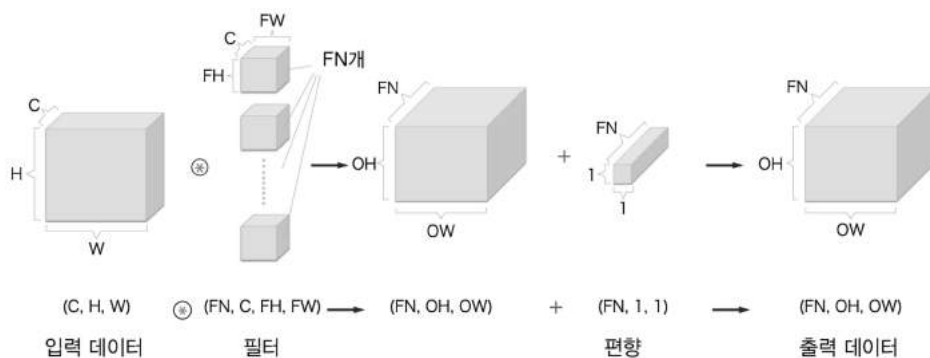
7-11그림

개의 feature map 을 출력하는 그림

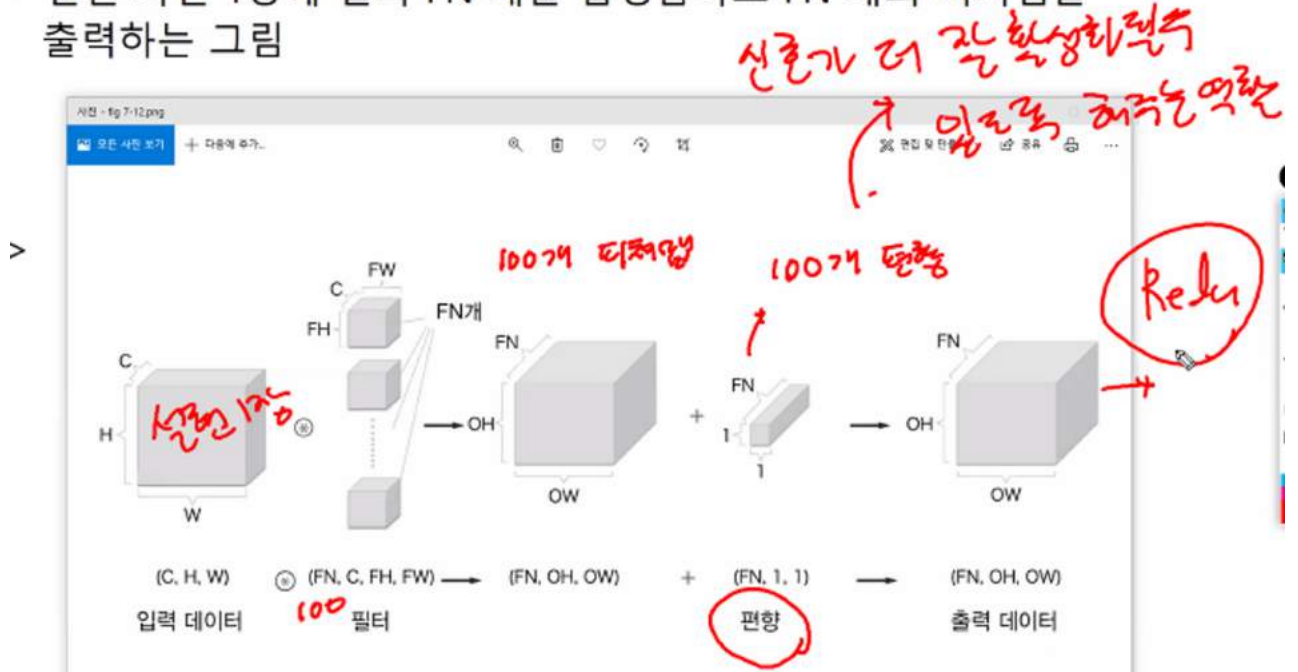


7-12 그림

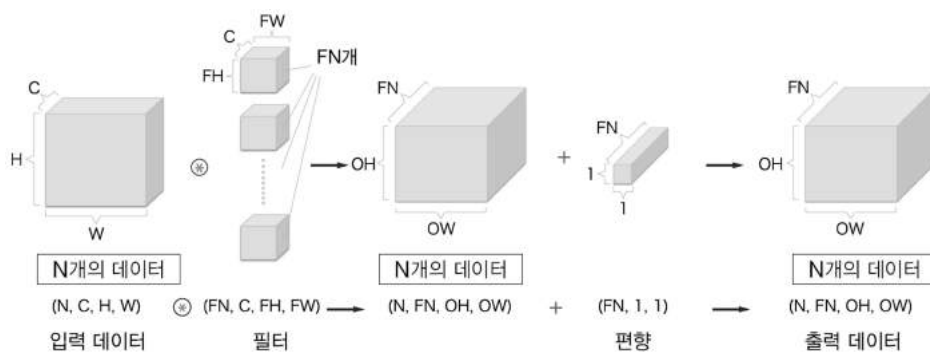
설현사진 1장에 필터 FN개를 합성곱하고 FN개의 피쳐맵을 출력하는 그림



설현 사진 1장에 필터 FN개를 합성곱하고 FN개의 피쳐맵을 출력하는 그림

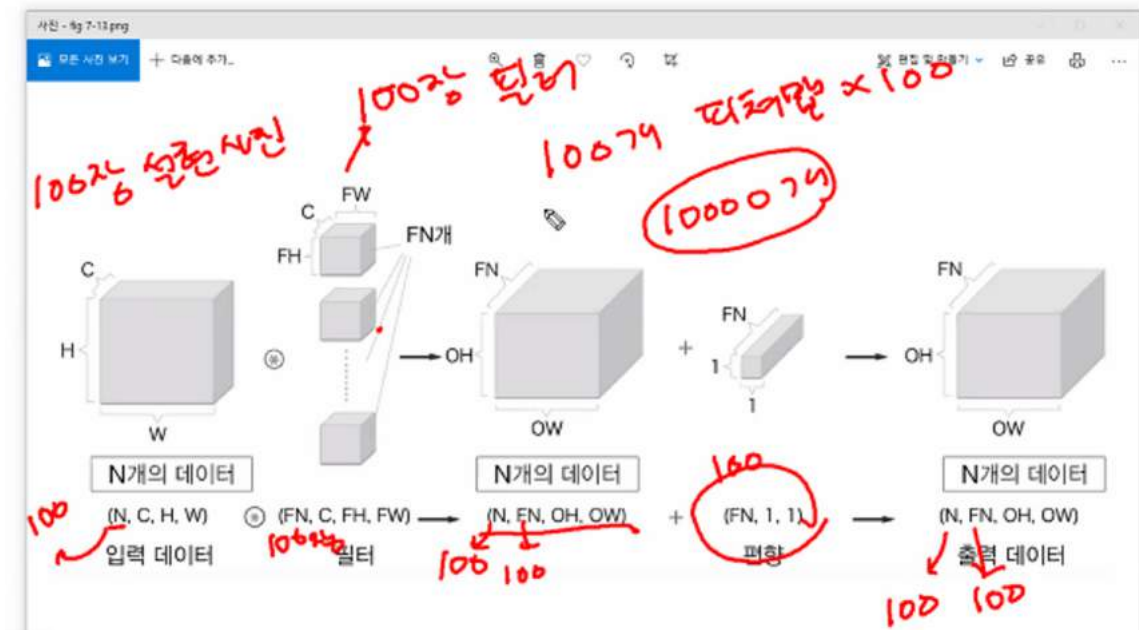


설현 사진 1장에 필터 FN개를 합성곱하고 FN개의 편향을 더해서 FN개의 피쳐맵을 출력하는 그림



## 7-13그림

한장씩 신경망에 넣고 학습을 시키면 너무 느리니까 100장씩 넣고 학습을 시키면 위와 같은 그림이 된다.



미니배치의 개수  $n$ 개만큼 설현 사진을 입력해서 필터 FN개와 합성곱 하여 미니배치 개수  $N$ 개만큼 피쳐맵을 출력하는 그림

정리 : RGB 설현 사진 1장을 100개의 RGB 필터로 합성곱 하면 100개의 피쳐맵이 생기는데 100개의 설현 사진이 100개의 RGB 필터와 각각 합성곱하면 설현사진 한장당 피쳐맵이 100개가 생성되므로 설현사진이 100장이니까 피쳐맵이 10000장이 된다.

## ■ 합성곱 계층을 이용하기 위한 im2col 함수를 사용하는 단계

im2col 함수가 convolution 층에서 사용되고 있는데 이 함수의 역할은?

배치단위의 원본이미지 -----> 신경망  
( 설현사진 100장 )  
↓  
4차원 ( 100, 3, 32, 32 )

4차원 데이터를 그대로 입력하면 컴퓨터가 합성곱 계산하는데 시간이 너무 많이 걸린다.  
시간을 줄이려면 어떻게 해줘야 하는가?

코드를 변경해줘야 한다. ( 함수 생성해주기 )

어제의 예로 들면 유럽 이미지를 합성곱해서 피쳐맵을 생성할 때 선생님이 짰 코드는 메모리 오류가 나면서 코드가 안돌았는데 세원씨가 짰 코드는 너무나 잘 수행되었다. 이것과 같은 것이다.

그래서 어떻게 코드를 변경하는가 ? 4차원을 2차원으로 변경/ 그리고 2차원으로 계산해 나가면 된다.

### 예제1. 설현 사진 한장을 4차원 행렬로 만드시오!

```
import numpy as np

x1 = np.random.rand( 1, 3, 7, 7 )
print(x1)
print(x1.shape)

print(x1.ndim) # 차원 확인, 4차원
```

### 예제2. 설현 사진 10장을 생성하시오!

```
import numpy as np

x1 = np.random.rand( 10, 3, 7, 7 )
print(x1)
print(x1.shape)

print(x1.ndim) #차원 확인, 4차원
```

### 예제3. 설현 사진 10장을 im2col 함수에 넣어서 2차원 행렬로 변환하시오!

4차원 -----> 2차원  
(10, 3, 7, 7)                      ( 90, 75 )

###im2col 함수코드 <https://cafe.daum.net/oracleoracle/Sedp/351>

```
def im2col(input_data, filter_h, filter_w, stride=1, pad=0):
    """다수의 이미지를 입력받아 2차원 배열로 변환한다(평탄화).

    Parameters
    -----
    input_data : 4차원 배열 형태의 입력 데이터(이미지 수, 채널 수, 높이, 너비)
    filter_h : 필터의 높이
    filter_w : 필터의 너비
    stride : 스트라이드
    pad : 패딩

    Returns
    -----
    col : 2차원 배열
    """
    N, C, H, W = input_data.shape
```

```

out_h = (H + 2 * pad - filter_h) // stride + 1
out_w = (W + 2 * pad - filter_w) // stride + 1

img = np.pad(input_data, [(0, 0), (0, 0), (pad, pad), (pad, pad)], 'constant')
col = np.zeros((N, C, filter_h, filter_w, out_h, out_w))

for y in range(filter_h):
    y_max = y + stride * out_h
    for x in range(filter_w):
        x_max = x + stride * out_w
        col[:, :, y, x, :, :] = img[:, :, y:y_max:stride, x:x_max:stride]

col = col.transpose(0, 4, 5, 1, 2, 3).reshape(N * out_h * out_w, -1)
return col

```

#####

```

import numpy as np

x1 = np.random.rand( 10, 3, 7, 7 )
col = im2col( x1, 5, 5, stride=1, pad=0 )
print( col.shape ) # (90, 75)

```

※ im2col 함수는 ?

신경망에서 합성곱을 진행하는데 입력되는 4차원 데이터를 2차원 데이터로 차원 축소해서 2차원 필터와 내적해서 합성곱하게 만드는 함수

model.add(Conv2D(32, (3, 3))) <--- 여기서 일어나는 일을 설명

↓

3x3의 필터 32개를 원본이미지( 100, 3, 32, 32)에 합성곱을 해서  
여러 개( 32x100개 )의 피쳐맵을 생성

im2col 함수는 4차원 ---> 2차원으로 변경하는 역할을 함

```

model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

```

**p 243 (그림 7-17)**



## 풀링층의 역할?



Convolution층이 이미지의 특징을 잡아내는 역할을 한다면 pooling층은 feature map 이미지를 선명하게 만드는 역할을 한다.

" 말 그대로 출력값에서 일부분만 취하는 기능 "

마치 사진을 축소하면 해상도가 좋아지는 듯한 효과와 비슷하다.

### \* 풀링( Pooling )의 종류 3가지 ?

1. 최대풀링 : 컨볼루션 데이터에서 가장 큰 값을 대표값으로 선정  
효과 : 이미지를 선명하게 만드는 효과
2. 평균풀링 : 컨볼루션 데이터에서 모든 값의 평균값을 대표값으로 선정  
효과 : 이미지를 부드럽게 하는 효과
3. 확률적 풀링 : 컨볼루션 데이터에서 임의 확률로 한개를 선정  
효과 : 우연히 이미지가 원본이미지와 같이 나타남

### 예제1. 파이썬으로 아래의 행렬을 만드시오!

```
21  8  8 12
12 19  9  7
 8 10  4  3
18 12  9 10
```

```
x= np.array([21, 8, 8, 12, 12, 19, 9, 7, 8, 10, 4, 3, 18, 12, 9, 10]).reshape(4,4)
print(x)
```

### 예제2. 아래의 행렬에서 max pooling을 시도해서 아래의 결과를 출력하시오!

```
21  8  8 12
12 19  9  7
 8 10  4  3
18 12  9 10
```

----->

```
21 12
18 10
```

```
import numpy as np
```

```
x = np.array([21, 8, 8, 12, 12, 19, 9, 7, 8, 10, 4, 3, 18, 12, 9, 10]).reshape(4,4)
```

```
a = im2col( x, 2, 2, 2, 0 )
           ↑   ↑   ↑   ↑
        필터가로, 필터세로, 스트라이드, 패딩
```

```
a = im2col( x2, 2, 2, 2, 0 )
d = np.max(a, axis=1)
print(d)
print(d.reshape(2,2))
```

## ■ 설현과 수지 사진 분류 신경망 만들기

지금 받은 **resize** 폴더를 **c드라이브** 밑에 **gimages2**에 둔다.

지금 받은 **loader9.py**를 **쥬피터** 워킹 디렉토리( 홈디렉토리 )에 가져다 둔다.

사진은 **1~600** : 설현 사진, **601~1230** : 수지사진

**loader9.py** 에는 **image\_load** 함수와 **label\_load** 함수가 있다.

```
import os
import re
import cv2
import numpy as np
import csv
import random
```

**# 1. 데이터를 로드한다.**

```
train_image= 'C:\\gimages2\\resize\\'
train_label = 'C:\\gimages2\\train_label.csv'
```

**# 2. 필요한 모듈을 임포트 한다.**

```
import loader9
from tensorflow.keras.models import Sequential, save_model
from tensorflow.keras.layers import Dense, Flatten, Dropout, BatchNormalization, Conv2D,
MaxPooling2D, Activation
from sklearn.model_selection import train_test_split
```

**# 3. 하이퍼 파라미터를 설정한다.**

```
batch_size = 28
num_classes = 2
epochs = 100
```

**# 4. 데이터를 로드 한다.**

```
x_train = loader9.image_load(train_image) # 이미지를 numpy array형 숫자로 변환
y_train = loader9.label_load(train_label) # csv파일을 읽어와서 one hot encoding한다.
#x_test = loader9.image_load(test_image)
```

```

#y_test = loader9.label_load(test_label)

print ( loader9.image_load(train_image).shape )
#print ( loader9.image_load(test_image).shape )
print ( loader9.label_load(train_label).shape)
#print ( loader9.label_load(test_label).shape )

#(x_train, y_train), (x_test, y_test) = cifar10.load_data()
# One hot Encoding
#y_train = np_utils.to_categorical(y_train)
#y_test = np_utils.to_categorical(y_test)

# 훈련과 테스트 데이터를 7대 3으로 나눔
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train,
                                                    test_size = 0.3,
                                                    random_state = 777)

# 모델 구성
model = Sequential()
# 첫번째 층은 반드시 input_shape 를 지정해야한다.
# x_train.shape ( 19000, 32, 32, 3 )
# input_shape=x_train.shape[1:] (32, 32, 3)
# mnist 와 비교하면 mnist 의 경우는 첫층이 완전연결계층이므로
# Flatten 시켜서 Flatten(input_shape=(28,28) ) 이렇게 입력했었고
# Convolution 층은 이미지의 형상이 무시되지 않도록 입력해야하기 때문에
# (128, 128, 3) 이렇게 3차원으로 넣어야한다.

#convolution 1층
model.add(Conv2D(32, (3, 3), padding='same', input_shape=x_train.shape[1:]))
model.add(BatchNormalization())
model.add(Activation('relu'))

#convolution 2층
model.add(Conv2D(32, (3, 3)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

# convolution 3층
model.add(Conv2D(64, (3, 3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))

# convolution 4층
model.add(Conv2D(64, (3, 3)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

```

```
model.add(Dropout(0.25))
```

## # 완전연결계층 2층

`model.add(Flatten())` # 완전 연결 계층에 입력할때는 flatten 시켜야 한다.

`model.add(Dense(512))` # 완전 연결계층 1층의 뉴런의 개수를 512개로 한다.

```
model.add(BatchNormalization())
```

```
model.add(Activation('relu'))
```

```
model.add(Dropout(0.5))
```

## #출력층

```
model.add(Dense(num_classes)) # 출력층의 뉴런의 개수 2개( 수지와 설현 분리므로)
```

```
model.add(Activation('softmax'))
```

## # 모델 설정

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
```

## # 데이터 정규화

```
x_train = x_train.astype('float32') # 실수형으로 숫자를 다 변환시킨다.
```

```
#x_test = x_test.astype('float32')
```

x\_train /= 255 # 255으로 숫자를 다 나눠서 0과 1사이의 숫자로 변환

```
#x_test /= 255
```

## # 모델 학습

```
hist = model.fit(x_train, y_train, validation_data=(x_val, y_val), epochs=epochs,
```

batch\_size=batch\_size)

## # 모델 평가

```
scores = model.evaluate(x_train, y_train, verbose=0)
```

```
#[0.3352026641368866, 0.9929999709129333]
```

#	오차	정확도
---	----	-----

```
print("CNN Error: %.2f%%" % (100-scores[1]*100))
```

```
save_model(model, "C:\wwgimages2\wwleaf9.h5") #모델을 저장, pickle파일 생성
```

03/22

2021년 3월 22일 월요일 오전 9:58

## ■ 구글 코랩을 이용해서 GPU를 맘껏 사용하면서 이파리 데이터를 분류해보기

1. 구글 코랩에 가입을 한다. ( 구글 검색창에서 colab이라고 검색한다. )
2. 새노트를 열고 GPU를 사용할 수 있도록 설정한다.
3. 카페 게시판에서 2개의 파일을 내려받는다.

[최종 포트폴리오 준비8] 코랩에서 이파리 데이터 분류 신경망 만들기

1. 이파리분류.ipynb
2. loader\_leaf.py

4. 이파리분류.ipynb 노트를 코랩에서 연다.

## ■ 신경망 코드를 이해하는 단계

1. tensorflow2.x에서 mnist 신경망

### # 1. 필요한 패키지를 가져오는 코드

```
import tensorflow as tf # 텐서 플로우 2.0
from tensorflow.keras.datasets.mnist import load_data #텐서플로우에 내장 되어 있는 mnist
# 데이터를 가져온다.

from tensorflow.keras.models import Sequential # 모델을 구성하기 위한 모듈
from tensorflow.keras.layers import Dense # 완전 연결계층을 구성하기 위한 모듈
from tensorflow.keras.utils import to_categorical # onehot encoding 하는 모듈

from sklearn.model_selection import train_test_split # 훈련과 검증 데이터 분리하는 모듈
tf.random.set_seed(777)
```

```
(x_train, y_train), (x_test, y_test) = load_data(path='mnist.npz') # mnist 데이터 로드
# 훈련 데이터 6만장, 테스트 데이터 1만장
```

**# 훈련 데이터를 가지고 훈련과 검정 데이터로 분리( 7:3 )**

```
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train,  
                                                  test_size = 0.3,  
                                                  random_state = 777)
```

```
print(x_train.shape) #(42000, 28, 28)
```

```
print(x_val.shape) # (18000, 28, 28)
```

**# 데이터를 정규화하는 코드**

```
num_x_train = x_train.shape[0] # 42000
```

```
num_x_val = x_val.shape[0] # 18000
```

```
num_x_test = x_test.shape[0] # 10000
```

**# 3차원 --> 2차원으로 차원축소하고서 정규화 진행 ( 한 픽셀이 0~255로 되어있음 )**

**# 0 ~ 1사이로 변경**

```
x_train = (x_train.reshape((num_x_train, 28 * 28))) / 255
```

```
x_val = (x_val.reshape((num_x_val, 28 * 28))) / 255
```

```
x_test = (x_test.reshape((num_x_test, 28 * 28))) / 255
```

**# 위의 정규화 작업 한것과 안한것과의 차이가 아주 크다. ( 학습할 때 나타남 )**

**# 하나의 숫자를 one hot encoding한다. ( 예 : 4 --> 0 0 0 0 1 0 0 0 0 )**

```
y_train = to_categorical(y_train) # 훈련 데이터의 라벨( 정답 )을 원핫 인코딩
```

```
y_val = to_categorical(y_val) # 검정 데이터의 라벨( 정답 )을 원핫 인코딩
```

```
y_test = to_categorical(y_test) # 테스트 데이터의 라벨( 정답 )을 원핫 인코딩
```

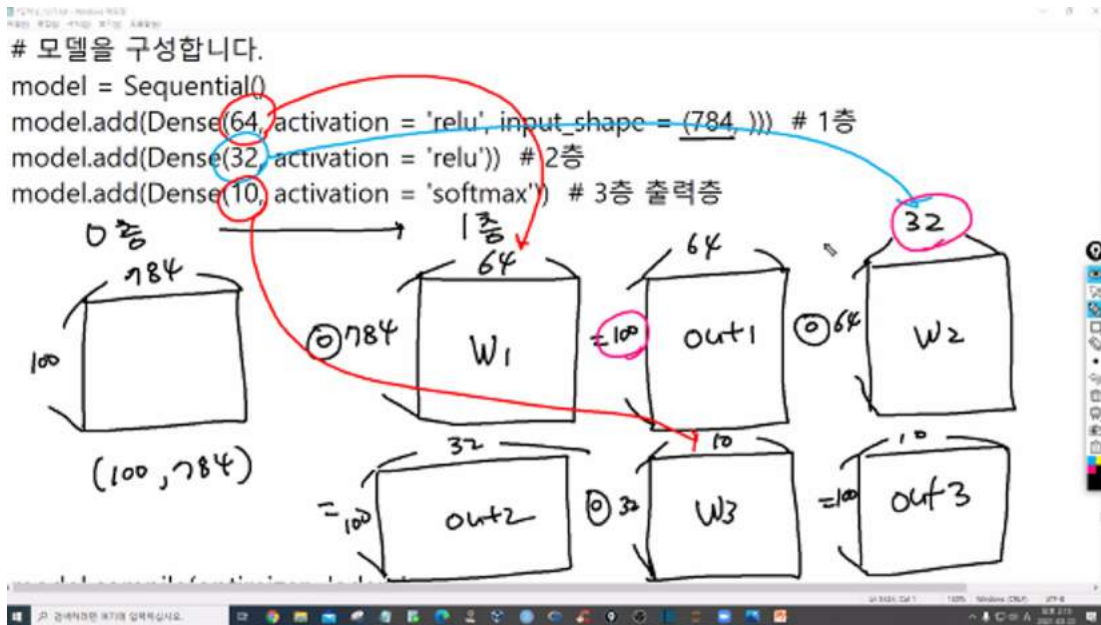
**# 모델을 구성한다.**

```
model = Sequential()
```

```
model.add(Dense(64, activation = 'relu', input_shape = (784, ))) # 1층
```

```
model.add(Dense(32, activation = 'relu')) # 2층
```

```
model.add(Dense(10, activation = 'softmax')) # 3층 출력층
```



# 모델을 설정한다. ( 경사하강법, 오차함수를 정의해줌 )

```

model.compile(optimizer='adam',
              loss = 'categorical_crossentropy',
              metrics=['acc']) # 학습과정에서 정확도를 보기 위함

```

# 모델을 훈련시킨다.

```

history = model.fit(x_train, y_train,
                    epochs = 30, # 30에폭
                    batch_size = 100,
                    validation_data = (x_val, y_val))

```

# 모델을 평가한다. ( 오차, 정확도가 출력된다. )

```

model.evaluate(x_test, y_test)

```

# 테스트 데이터의 예측값을 출력한다.

```

results = model.predict(x_test)

```

#정확도를 확인한다.

```

y_hat = np.argmax( results, axis = 1 )
print( np.sum( y_hat == y_test ) / len( y_test ) )

```

# 시각화

```

import matplotlib.pyplot as plt

```

```

his_dict = history.history
loss = his_dict['loss']

```

val\_loss = his\_dict['val\_loss'] # 검증 데이터가 있는 경우 'val\_' 수식어가 붙습니다.

```

epochs = range(1, len(loss) + 1)
fig = plt.figure(figsize = (10, 5))

```

### # 훈련 및 검증 손실 그리기

```
ax1 = fig.add_subplot(1, 2, 1)
ax1.plot(epochs, loss, color = 'blue', label = 'train_loss')
ax1.plot(epochs, val_loss, color = 'orange', label = 'val_loss')
ax1.set_title('train and val loss')
ax1.set_xlabel('epochs')
ax1.set_ylabel('loss')
ax1.legend()
```

```
acc = his_dict['acc']
val_acc = his_dict['val_acc']
```

### # 훈련 및 검증 정확도 그리기

```
ax2 = fig.add_subplot(1, 2, 2)
ax2.plot(epochs, acc, color = 'blue', label = 'train_loss')
ax2.plot(epochs, val_acc, color = 'orange', label = 'val_loss')
ax2.set_title('train and val loss')
ax2.set_xlabel('epochs')
ax2.set_ylabel('loss')
ax2.legend()
```

```
plt.show()
```

## 1. mnist 데이터 cnn 안 썼을 때 0.97의 정확도

### ■ cnn 사용 안 했을 때의 코드

```
from tensorflow.keras.datasets.fashion_mnist import load_data
from tensorflow.keras.utils import to_categorical
```

```
# Fashion-MNIST 데이터를 다운받습니다.
```

```
(x_train, y_train), (x_test, y_test) = load_data()
```

```
# 신경망 데이터 넣기전에 데이터 정규화( 값의 범위를 0~1 로 만들어 줍니다.)
```

```
x_train = x_train / 255
```

```
x_test = x_test / 255
```

```
# 라벨을 one hot encoding 해준다.
```

```
y_train = to_categorical(y_train)
```

```
y_test = to_categorical(y_test)
```

```
# 훈련 데이터에 대한 검정을 위해 validation 데이터를 만든다.
```

```
# 테스트 데이터는 나중에 시험볼때 사용할 것이므로 그냥 두고
```

```
# 훈련 데이터만 7대3 으로 분리 합니다.
```

```
import tensorflow as tf
```

```
from sklearn.model_selection import train_test_split
```



```
tf.random.set_seed(777)
```

```
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size = 0.3, random_state = 777)
```

```
# 모델을 구성한다.
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
```

```
model = Sequential()
model.add( Flatten(input_shape =(28,28)) ) # 완전연결계층으로 데이터 입력
model.add(Dense(64, activation = 'relu')) # 64개의 뉴런을 가진 1층
model.add(Dense(32, activation = 'relu')) # 32개의 뉴런을 가진 2층
model.add(Dense(10, activation = 'softmax')) # 10개 뉴런을 가진 출력층
```

```
# 학습과정 설정하기
```

```
model.compile(optimizer='adam',
              loss = 'categorical_crossentropy',
              metrics=['acc']) # 모니터링할 평가지표 : acc
```

```
# 모델 학습하기
```

```
history = model.fit(x_train, y_train,
                    epochs = 30,
                    batch_size = 100,
                    validation_data = (x_val, y_val))
```

```
# 모델 평가하기
```

```
model.evaluate(x_test, y_test)
```

```
# 테스트 데이터의 예측값을 출력한다.
```

```
results = model.predict(x_test)
```

```
# 정확도 출력
```

```
import numpy as np
y_hat = np.argmax( results, axis = 1 )
y_label = np.argmax( y_test, axis = 1 )
print( np.sum( y_hat == y_label ) / len( y_test ) )
```

```
# 시각화
```

```
import matplotlib.pyplot as plt
```

```
his_dict = history.history
loss = his_dict['loss']
val_loss = his_dict['val_loss'] # 검증 데이터가 있는 경우 'val_' 수식어가 붙습니다.
```

```
epochs = range(1, len(loss) + 1)
```

```

fig = plt.figure(figsize = (10, 5))

# 훈련 및 검증 손실 그리기
ax1 = fig.add_subplot(1, 2, 1)
ax1.plot(epochs, loss, color = 'blue', label = 'train_loss')
ax1.plot(epochs, val_loss, color = 'orange', label = 'val_loss')
ax1.set_title('train and val loss')
ax1.set_xlabel('epochs')
ax1.set_ylabel('loss')
ax1.legend()

acc = his_dict['acc']
val_acc = his_dict['val_acc']

# 훈련 및 검증 정확도 그리기
ax2 = fig.add_subplot(1, 2, 2)
ax2.plot(epochs, acc, color = 'blue', label = 'train_loss')
ax2.plot(epochs, val_acc, color = 'orange', label = 'val_loss')
ax2.set_title('train and val loss')
ax2.set_xlabel('epochs')
ax2.set_ylabel('loss')
ax2.legend()

plt.show()

```

## ■ cnn 쓰는 것

# 1. 필요한 모듈을 로드한다.

```

import tensorflow as tf

from tensorflow.keras.utils import to_categorical
from tensorflow.keras.datasets.fashion_mnist import load_data
from tensorflow.keras.models import Sequential, save_model
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Conv2D, MaxPooling2D # cnn 구현에 필요한 모듈 임포트

```

# Fashion-MNIST 데이터를 다운받습니다.

# 정규화를 진행한다.

```

(x_train, y_train), (x_test, y_test) = load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
print(x_train.shape) #(60000, 28, 28)
x_train = x_train.reshape(-1,28,28,1) # 3차원을 4차원으로 바꿔준다.

```

# cnn 층에 데이터를 입력할 때는 4차원으로 입력해줘야 하기 때문이다.

```

print(x_train.shape) #(60000, 28, 28, 1)

```



```

model.add( Dropout(0.2) )
model.add( Dense(32, activation='relu') ) # 완전연결계층 3층
model.add( Dropout(0.2) )
model.add( Dense(10, activation='softmax') ) # 출력층 4층


model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

hist = model.fit(x_train, y_train, batch_size=100 ,epochs=30, validation_data=(x_val ,y_val))

x_test = x_test.reshape(-1,28,28,1)
print(x_test.shape)

# 모델 평가하기
model.evaluate(x_test, y_test)
results = model.predict(x_test)

# 정확도 출력
import numpy as np
y_hat = np.argmax(results, axis=1)
y_label = np.argmax(y_test,axis=1)

print ( np.sum( y_hat == y_label ) / len(y_test) )


# 시각화
import matplotlib.pyplot as plt

his_dict = history.history
loss = his_dict['loss']
val_loss = his_dict['val_loss'] # 검증 데이터가 있는 경우 'val_' 수식어가 붙습니다.

epochs = range(1, len(loss) + 1)
fig = plt.figure(figsize = (10, 5))

# 훈련 및 검증 손실 그리기
ax1 = fig.add_subplot(1, 2, 1)
ax1.plot(epochs, loss, color = 'blue', label = 'train_loss')
ax1.plot(epochs, val_loss, color = 'orange', label = 'val_loss')
ax1.set_title('train and val loss')
ax1.set_xlabel('epochs')
ax1.set_ylabel('loss')
ax1.legend()

acc = his_dict['acc']
val_acc = his_dict['val_acc']

# 훈련 및 검증 정확도 그리기

```

```
ax2 = fig.add_subplot(1, 2, 2)
ax2.plot(epochs, acc, color = 'blue', label = 'train_loss')
ax2.plot(epochs, val_acc, color = 'orange', label = 'val_loss')
ax2.set_title('train and val loss')
ax2.set_xlabel('epochs')
ax2.set_ylabel('loss')
ax2.legend()
```

```
plt.show()
```

**문제. 아래의 코드를 참고해서 배치정규화 코드를 중간중간 삽입하고 ( relu층 앞에 준다. )  
에폭을 100에폭으로 돌리시오.**

```
model.add( Conv2D(64, ( 3, 3 ), padding = 'same' )
model.add( BatchNormalization() )
model.add( Activation('relu') )
```

03/23

2021년 3월 23일 화요일 오전 9:35

## ■ 밑바닥 딥러닝 복습

- 1장. numpy 사용법
- 2장. 퍼셉트론
- 3장. 3층 신경망 구현( 학습 x )
- 4장. 2층 신경망 구현( 학습 o ) - 수치미분을 이용
- 5장. 2층 신경망 구현( 학습 o ) - 오차역전파를 이용
- 6장. 언더피팅과 오버피팅을 방지하는 방법
- 7장. CNN 층
- 8장. 딥러닝 역사를 텐서플로우 2.0을 이용해서 구현해보기

### \* 딥러닝을 발전 시킨 유명한 신경망들

1. CNN을 사용한 신경망 구현 ( fashion mnist )
2. VGG 신경망

### \* 신경망을 사용할 때 불편해서 구현하게 된 기타 기능들

1. 이미지 증식 시키기
2. 케라스의 콜백 기능

### \* 신경망 활용 홈페이지 생성 ( 기본 포트폴리오 )

## ■ CNN을 사용한 신경망 구현( 수지와 설현사진 분류 )

Fashion mnist 전체코드를 가져와서 수정한다.

1. 제규어와 얼룩말 데이터 2000장( 한클래스당 1000장 )을 받고 코랩에 올리기
2. gpu를 사용하는지 확인한다.  
!nvidia-smi -L
3. drive를 마운트 시킨다.

```
from google.colab import drive
drive.mount('/content/drive')
```

4. 테스트데이터는 /content/gdrive/MyDrive/samples6/train\_resize2  
훈련데이터는 /content/gdrive/MyDrive/samples6/train\_resize2

### 터미널 여는법:

```
!pip install kora
from kora import console
console.start() # and click lin
```

5. 얼룩말과 제규어 사진을 가지고 홈페이지 구현을 위한 신경망 모델 생성하는 코드를 수행한다.

아래의 코드를 코랩에서 엽니다.

1. 제규어와 지브라.ipynb

## ■ 텐서 플로우2.x의 전의 학습 기능

전의학습은 사전에 학습된 네트워크의 가중치 또는 이미지넷 대회에서 우승한 유명한 신경망 모델을 가져와서 사용하는 기능을 말한다.

1. 모델을 변형하지 않고 그대로 사용하는 방법

예 : from keras.applications import VGG16

```
vgg16 = VGG16(weights = 'imagenet', input_shape = (32, 32, 3), include_top = False)
```

설명 : weights : imagenet 데이터를 학습시킨 가중치의 사용여부를 결정한다.

기본값은 None ( 1000개의 이미지 사진들이 있고 그 사진들을 학습시킨 가중치 )

input\_shape = 입력데이터의 사이즈를 기술한다.

include\_top = 완전연결계층의 모델의 분류기를 내가 직접 기술할지 말지를 결정한다.

( False로 되어있으면 내가 직접 완전연결계층을 짜겠다 )

## ■ 텐서 플로우 2.0의 장점

1. 실행모드로 실행할 수 있다. ( 파이썬처럼 실행이 가능하다 )
2. 이미지넷 대회에서 우승한 유명한 신경망 모델 설계와 가중치 파일을 내가 분류하고자하는 신경망 코드에 쉽게 가져올 수 있다. ( 전이학습 )
3. 이미지를 증식 시키고 증식 시킨 이미지들을 쉽게 신경망에 입력할 수 있다.

4. EarlyStopping 기능을 구현할 수 있다.

## ■ EarlyStopping 기능

EarlyStopping 콜백을 직역하면 '이른 멈춤'이다. 즉 모델 학습시 지정된 기간동안 모니터링하는 평가지표에서 성능향상이 일어나지 않은 경우 학습을 중단한다. 주로 많이 사용하는 콜백인자는 다음과 같다.

**예제:**

```
EarlyStopping( monitor='val_loss', patience=2, verbose=0, mode='auto')
```

monitor : 모니터링 할 평가지표를 설정한다. ( 오차: val\_loss, 정확도 : val\_acc )

verbose : 콜백의 수행과정 노출여부를 결정한다.

0 : 아무런 표시를 안하겠다.

1 : 프로그래스 바( progress bar )를 출력

2 : 에폭마다 수행과정을 출력

patience : 지정한 수만큼의 기간에서 평가지표의 향상이 일어나지 않을 경우 학습을 중단하겠다.

patience = 5 를 썼다면 5번은 참아주겠다는 뜻이다.

**구현 :**

```
callbacks = [ EarlyStopping( monitor='val_loss', patience=2, verbose=0, mode='auto') ]
```

```
model.fit( x_train, y_train,
          batch_size = 32,
          validation_data=( x_val, y_val ),
          epochs = 10,
          callbacks = callbacks )
```

```
model.summary()
```

```
from keras.callbacks import ModelCheckpoint, EarlyStopping
```

```
filepath = 're_w2.h5'
```

```
callbacks = [ ModelCheckpoint( filepath = filepath , monitor = 'val_loss' , verbose=1,
save_best_only=True ) ]
```

훈련 데이터는 정확도 0.99, 검증 데이터와 테스트 데이터의 정확도 : 0.91

## ■ R shiny를 이용해서 신경망을 쉽게 이용하는 방법



카페 글 285번 참조

<https://cafe.daum.net/oracleoracle/SgRM/285>

# 검증 데이터셋을 만듭니다.

```
from sklearn.model_selection import train_test_split
np.random.seed(111)
```

# 훈련/테스트 데이터를 0.7/0.3의 비율로 분리합니다.

```
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train,
                                                  test_size = 0.2, random_state = 777)
```

```
setwd("D:\\W\\Wys267")
```

```
packages <- c('imager', 'shiny', 'jpeg', 'png', 'reticulate', 'devtools')
```

```
if (length(setdiff(packages, rownames(installed.packages()))) > 0) {
  install.packages(setdiff(packages, rownames(installed.packagesvx ())))
}
```

```
if (length(setdiff("keras", rownames(installed.packages()))) > 0) {
  devtools::install_github("rstudio/keras")
}
```

```
require(imager)
require(shiny)
require(jpeg)
require(png)
library(reticulate)
library(keras)
```

```
#setwd(tempfile())
#setwd("/Users/aiden/Desktop/data/cifar10_densenet")
```

```
load("envir.RData")
model <- load_model_hdf5("re_w3.h5")
```

```
synsets <- readLines("synset.txt")
```

```
server <- shinyServer(function(input, output) {
  ntext <- eventReactive(input$goButton, {
    print(input$url)
    if (input$url == "http://") {
      NULL
    } else {
      tmp_file <- tempfile()
    }
  })
})
```

```

        download.file(input$url, destfile = tmp_file, mode = 'wb')
        tmp_file
    }
})

```

```

output$originImage = renderImage({
  list(src = if (input$tabs == "Upload Image") {
    if (is.null(input$file1)) {
      if (input$goButton == 0 || is.null(ntext())) {
        '904_google.jpg'
      } else {
        ntext()
      }
    } else {
      input$file1$datapath
    }
  } else {
    if (input$goButton == 0 || is.null(ntext())) {
      if (is.null(input$file1)) {
        '904_google.jpg'
      } else {
        input$file1$datapath
      }
    } else {
      ntext()
    }
  },
  title = "Original Image")
}, deleteFile = FALSE)

```

```

output$res <- renderText({
  src = if (input$tabs == "Upload Image") {
    if (is.null(input$file1)) {
      if (input$goButton == 0 || is.null(ntext())) {
        '904_google.jpg'
      } else {
        ntext()
      }
    } else {
      input$file1$datapath
    }
  } else {
    if (input$goButton == 0 || is.null(ntext())) {
      if (is.null(input$file1)) {
        '904_google.jpg'
      } else {
        input$file1$datapath
      }
    } else {
      ntext()
    }
  }
}

```

```

img <- load.image(src)
plot(img)
img <- image_load(src, target_size = c(32,32))
img
x <- image_to_array(img)
# ensure we have a 4d tensor with single element in the batch dimension,
x <- array_reshape(x, c(1, dim(x)))

# normalize
x[,,,1] <- x[,,,1] / 255
x[,,,2] <- x[,,,2] / 255
x[,,,3] <- x[,,,3] / 255

# predict
preds <- model %>% predict(x)

# output result as string
max.idx <- order(preds[1,], decreasing = TRUE)[1]
result <- synsets[max.idx]
res_str <- ""
tmp <- strsplit(result[1], " ")[[1]]
res_str <- paste0(res_str, tmp[2])
res_str
})
})

```

```

require(imager)
require(shiny)
require(jpeg)
require(png)

ui <- shinyUI(
  fluidPage(
    includeCSS("bootstrap.css"),

    pageWithSidebar(
      headerPanel(title = '수지설현 using DenseNet',
        windowTitle = 'Image Classification(수지설현) using DenseNet'),

      fluidRow(
        column(1,
          column(9,
            tabsetPanel(
              id = "tabs",
              tabPanel("Upload Image",
                fileInput('file1', 'Upload a PNG / JPEG File:')),
              tabPanel(
                "Use the URL",

```

```

        textInput("url", "Image URL:", "http://"),
        actionButton("goButton", "Go!")
      )
    ),
    h3(titlePanel("DESCRIPTION - 수지설현 분류")),
    h3(titlePanel("수지와 설현"))

  ),
  column(2)
),

mainPanel(
  h3("Image"),
  tags$hr(),
  imageOutput("originImage", height = "auto"),
  tags$hr(),
  h3("What is this?"),
  tags$hr(),
  verbatimTextOutput("res")
)

)))

shinyApp(ui = ui, server = server)

```

# 03/24

2021년 3월 24일 수요일 오전 10:06

## ■ R

```
##### set this file location to working directory
#####
packages <- 'rstudioapi'
if (length(setdiff(packages, rownames(installed.packages()))) > 0) {
  install.packages(setdiff(packages, rownames(installed.packages())))
}
library('rstudioapi')
current_dir<-dirname(rstudioapi::getSourceEditorContext())$path)
setwd(current_dir)
```

```
# package_in : 없으면 설치, 있으면 library화 하라는 함수
package_in<-function(p_name,option=1){
  packages <- p_name
  if (length(setdiff(packages, rownames(installed.packages()))) > 0) {
    install.packages(setdiff(packages, rownames(installed.packages())))
  }
  if (option==1){
    library(p_name,character.only = TRUE)
  }
}
```

```
#####1. 패키지 설치
#####
```

```
package_in('shinydashboard') # shinydashboard 패키지가 없으면 설치하고 있으면 library
화 하라는 뜻
package_in('shiny')
package_in('ggplot2')
package_in('plotly')
package_in('lattice')
```

```
##### 2. 화면 개발
#####
```

```
sidebar <- dashboardSidebar(
  sidebarMenu(
    # fileInput : csv 파일 및 다양한 파일들을 불러오는 화면 구현
    fileInput("file1", "Choose CSV File",
      multiple = FALSE,
      accept = c("text/csv",".xlsx",".txt",
        "text/comma-separated-values,text/plain",
        ".csv")),
```

```

# 그래프 종류 보여줌
menuItem("Plot",
      menuSubItem('막대그래프',tabName='barplot'), #이름
      menuSubItem('원형그래프',tabName='piechart')

)

)

)

body <- dashboardBody(

  tabItems(

    ##### bar plot
    tabItem(tabName = "barplot",
      sidebarPanel(
        selectInput("in_sel_bar_yVar","y Variable:", choices = NULL),
        selectInput("in_sel_bar_xVar","x Variable:", choices = NULL)
      ),
      mainPanel(
        plotOutput('plot_bar')
      )
    ),
    ##### piechart
    tabItem(tabName = "piechart",
      sidebarPanel(
        selectInput("in_sel_pie_xVar","x Variable:", choices = NULL)
      ),
      mainPanel(
        plotlyOutput('plot_pie')
      )
    )
  )
)

ui<-dashboardPage(
  dashboardHeader(title='my graph'),
  sidebar,
  body

)

```

```
#####3. 서버단 개발
#####
```

```
server <- function(input, output,session) {
  options(warn = -1)
  options(shiny.maxRequestSize = 30*1024^2)

  dataload<-reactive({
    # 서버로 해당 파일 불러옴
    req(input$file1)

    # 파일이름을 file1 변수에 넣음
    file1 = input$file1
    # 파일이름.csv가 불러와져서 data1에 로드됨
    data1 = read.csv(file1$datapath)

    updateSelectInput(session, "in_sel_bar_xVar", choices = colnames(data1))
    updateSelectInput(session, "in_sel_bar_yVar", choices = colnames(data1))

    updateSelectInput(session, "in_sel_pie_xVar", choices = data1[,1])

    return(data1)

  })

  #####nomal_bar
  output$plot_bar <- renderPlot({
    table_in<-dataload()

    xdata<-as.factor(table_in[,input$in_sel_bar_xVar])
    ydata<-as.factor(table_in[,input$in_sel_bar_yVar])
    fdata=data.frame(x=xdata,y=ydata)

    ggplot(fdata) +
      geom_bar(aes_string(x='x',y='y',fill='x'),stat = "identity",show.legend=F)

  })

  output$plot_pie <- renderPlotly({
    table_in<-dataload()

    plot_ly(table_in, labels = ~colnames(table_in)[-1],
    values=~as.factor( table_in[table_in[,1] == input$in_sel_pie_xVar,-1] ),type='pie')
```

```

    })
  }

##### 4. 샤이니 실행
#####

shinyApp(ui = ui, server = server)

```

## ■ 홈페이지 구현을 위한 샤이니 ui 의 기능들을 모아놓은 코드

```

library(shiny)

# Define UI ----
ui <- fluidPage(
  titlePanel("Basic widgets"),

  fluidRow(

    column(3,
      h3("Buttons"),
      actionButton("action", "Action"),
      br(),
      br(),
      submitButton("Submit")),

    column(3,
      h3("Single checkbox"),
      checkboxInput("checkbox", "Choice A", value = TRUE)),

    column(3,
      checkboxGroupInput("checkGroup",
        h3("Checkbox group"),
        choices = list("Choice 1" = 1,
          "Choice 2" = 2,
          "Choice 3" = 3),
        selected = 1)),
  )
)

```



```
column(3,  
  dateInput("date",  
    h3("Date input"),  
    value = "2014-01-01"))  
,
```

```
fluidRow(  
  

```

```
column(3,  
  dateRangeInput("dates", h3("Date range"))),
```

```
column(3,  
  fileInput("file", h3("File input"))),
```

```
column(3,  
  h3("Help text"),  
  helpText("Note: help text isn't a true widget",  
    "but it provides an easy way to add text to",  
    "accompany other widgets.")),
```

```
column(3,  
  numericInput("num",  
    h3("Numeric input"),  
    value = 1))  
,
```

```
fluidRow(  
  

```

```
column(3,  
  radioButtons("radio", h3("Radio buttons"),  
    choices = list("Choice 1" = 1, "Choice 2" = 2,  
      "Choice 3" = 3),selected = 1)),
```

```
column(3,  
  selectInput("select", h3("Select box"),  
    choices = list("Choice 1" = 1, "Choice 2" = 2,
```

```

"Choice 3" = 3), selected = 1)),

column(3,
  sliderInput("slider1", h3("Sliders"),
    min = 0, max = 100, value = 50),
  sliderInput("slider2", "",
    min = 0, max = 100, value = c(25, 75))
),

column(3,
  textInput("text", h3("Text input"),
    value = "Enter text..."))
)

)

# Define server logic ----
server <- function(input, output) {

}

# Run the app ----
shinyApp(ui = ui, server = server)

```

## ■ 홈페이지 구현

<https://cafe.daum.net/oracleoracle/SgRM/294>

참고

```

library(rsconnect)

rsconnect::setAccountInfo(name='theworldbest',
  token='11BBEFA03AB628537F49E34A470285E0',
  secret='lxPVzj1EUUxMvwz8kgIWlwrINmFNoEfrRQoMFwfX')

rsconnect::deployApp('d:\www\mys277',appName = "myapp2777")

```

데이터 분석가

<https://theworldbest.shinyapps.io/myapp2777/> 데이터분석 그래프 시각화 홈페이지 구축

## **2. 포트폴리오 2**

머신러닝( IBM의 왓슨 )

## **3. 포트폴리오 3**

딥러닝 활용 신경망 포트폴리오( 딥러닝 개발자, 연구원 )

03/25

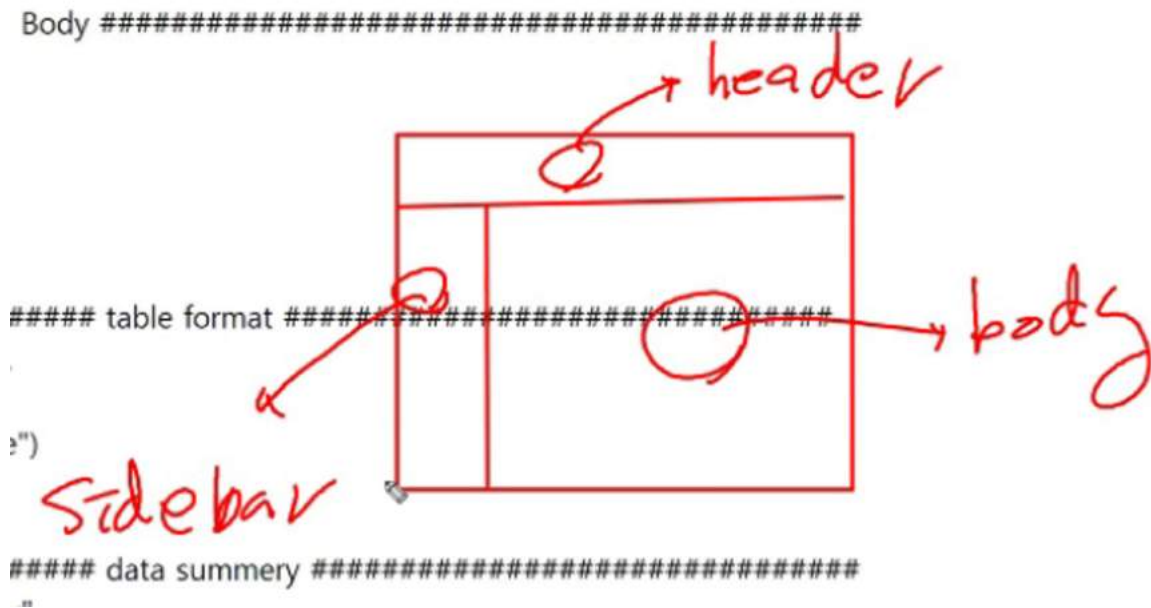
2021년 3월 25일 목요일 오전 9:45

## 2. 포트폴리오 2

R shiny를 이용해서 머신러닝( IBM의 왓슨 ) 데이터 분석을 편하게 하기

하루에 만원씩 벌 수 있는 코드

```
Body #####
```



```
##### table format #####
```

```
##### data summary #####
```

```
##### set this file location to working directory
#####
```

```
packages <- 'rstudioapi'
if (length(setdiff(packages, rownames(installed.packages()))) > 0) {
  install.packages(setdiff(packages, rownames(installed.packages())))
}
library('rstudioapi')
current_dir<-dirname(rstudioapi::getSourceEditorContext())$path)
```

```
package_in<-function(p_name,option=1){
  packages <- p_name
  if (length(setdiff(packages, rownames(installed.packages()))) > 0) {
    install.packages(setdiff(packages, rownames(installed.packages())))
  }
  if (option==1){
    library(p_name,character.only = TRUE)
  }
}
```

## ##### 1. 패키지 설치 #####

```
package_in('shinydashboard')
package_in('shinydashboardPlus')
package_in('shiny')
```

```
#### data load
package_in('DT')
```

```
#### summary
package_in('data.table')
package_in('grid')
package_in('ggplot2')
```

```
#### decision tree
package_in('C50')
package_in('gmodels')
```

```
#### JRip
#package_in('RWeka')
```

```
#### knn
package_in('class')
package_in('gmodels')
```

```
#### naive
package_in("e1071")
package_in("data.table")
```

```
#### neuralnet
package_in('neuralnet')
```

```
#### graph
package_in('ggplot2')
package_in('dplyr')
package_in('plotly')
package_in('lattice')
package_in('GGally')
```

## ##### 2. 화면 개발

```
#####
##### 2-1. Header
#####
```

```
header <- dashboardHeader(
  title='R Project for yys'
)
```

```
##### 2-2. Sidebar
#####
```

```

sidebar <- dashboardSidebar(
  fileInput("file1", "Choose CSV File",
    multiple = FALSE,
    accept = c("text/csv", ".xlsx", ".txt",
      "text/comma-separated-values,text/plain",
      ".csv") ),
  sidebarMenu(
    menuItem("Table",icon=icon('table'),
      menuSubItem('Tableformat',tabName='tableformat') ),

    menuItem("Statistics",icon=icon('file-contract'),
      menuSubItem('Data Summary',tabName='datasummary'),
      menuSubItem('Regression',tabName = 'regression'),
      menuSubItem('Decision Tree',tabName = 'decision')),
    menuItem("Machine Learning",icon=icon('laptop'),
      menuSubItem('JRip',tabName = 'jrip'),
      menuSubItem('KMeans',tabName='kmeans'),
      menuSubItem('KNN',tabName='knn'),
      menuSubItem('Naive bayes',tabName = 'naive'),
      menuSubItem('Naive bayes',tabName = 'association'), #새로운 메뉴 추가 원할시
      menuSubItem('Neuralnet',tabName = 'neuralnet')),
    menuItem("Graph",icon=icon('chart-area'),
      menuSubItem('Barplot',tabName='barplot'),
      menuSubItem('Piechart',tabName='piechart'),
      menuSubItem('Lineplot',tabName='lineplot'),
      menuSubItem('Scatterplot',tabName='scatterplot'),
      menuSubItem('Boxplot',tabName='boxplot'),
      menuSubItem('Pairs',tabName='pairs'))

  )

)

```

```

##### 2-3. Body
#####
body <- dashboardBody(

```

```

  tabItems(

    ##### table format
    #####
    menuItem(tabName = "tableformat",
      mainPanel(
        DT::dataTableOutput("table")
      )
    ),

```

```

##### data summery
#####
tabItem(tabName = "datasummary",
  fluidRow(
    box(
      title = "Summary",  solidHeader = TRUE,
      collapsible = TRUE, width = 6,
      verbatimTextOutput("stat_summary")
    ),
    box(
      title = "Quantile",  solidHeader = TRUE,
      collapsible = TRUE, width = 6,
      verbatimTextOutput("quan")
    ),
    box(
      title = "Quantile Graph",  solidHeader = TRUE,
      collapsible = TRUE,width = 12,
      plotOutput("plot_quan",height='auto')
    ),
    box(
      title = "Distribution Graph",  solidHeader = TRUE,
      collapsible = TRUE,width = 12,
      plotOutput("plot_dist",height='auto')
    )
  )
),
##### regression
#####
tabItem(tabName = "regression",

  fluidRow(
    box(
      title = "Data",  solidHeader = TRUE,
      collapsible = TRUE,width = 9,
      verbatimTextOutput("submit_input_sample_regression")
    ),
    box(
      width=2,
      uiOutput("dependents_delcol_regression",
        style = "overflow-y:scroll;
        max-height: 100px; background:
        ghostwhite;"),
      uiOutput("dependents_selcol_regression"),

      uiOutput("dependents_button_regression")
    ),
    box(
      title = "ANOVA Table",  solidHeader = TRUE,
      collapsible = TRUE,width = 9,
      verbatimTextOutput("anova_Render_regression")
    )
  )
)

```

```

),
box(
  title = "Regression", solidHeader = TRUE,
  collapsible = TRUE,width = 9,
  verbatimTextOutput("coefficient_Render_regression")
),
box(
  title = "Parameter Confidence Interval", solidHeader = TRUE,
  collapsible = TRUE,width = 9,
  verbatimTextOutput("interval_Render_regression")
),
box(
  title = "Model Plot", solidHeader = TRUE,
  collapsible = TRUE,width = 9,
  plotOutput("plot_reg",height='auto')
)
),
),
##### decision
#####
tabItem(tabName = "decision",
  fluidRow(
    box(title = "Data", solidHeader = TRUE,
      collapsible = TRUE,width = 8,
      verbatimTextOutput("submit_input_sample_decision")
    ),
    box(
      width=2,
      uiOutput("dependents_delcol_decision",
        style = "overflow-y:scroll; max-height: 100px;
        background: ghostwhite;")
    ),
    box(
      width=2,
      uiOutput("dependents_selcol_decision"),
      uiOutput("dependents_selmodel_decision"),
      uiOutput("dependents_selwinnow_decision"),
      numericInput("trials", "Trials:", 1, min = 1),
      numericInput("seed", "Seed:", 1),
      uiOutput("dependents_button_decision")
    ),
    box(title = "Cross Table", solidHeader = TRUE,
      collapsible = TRUE,width = 9,
      verbatimTextOutput("TestTableRender_decision")
    ),
    box(title = "Model Summary", solidHeader = TRUE,
      collapsible = TRUE,width = 9,
      verbatimTextOutput("TestSummaryRender_decision")
    )
  )
),
),
##### Jrip #####

```



```

tabItem(tabName = "jrip",
  fluidRow(
    box(title = "Data", solidHeader = TRUE,
      collapsible = TRUE,width = 9,
      verbatimTextOutput("submit_input_sample_jrip")
    ),
    box(width=2,
      uiOutput("dependents_delcol_jrip",
        style = "overflow-y:scroll; max-height: 100px;
        background: ghostwhite;"),
      uiOutput("dependents_selcol_jrip"),
      numericInput("numopt", "NumOpt", 1, min = 1),
      numericInput("seed", "Seed:", 1),
      uiOutput("dependents_button_jrip")
    ),
    box(title = "Cross Table", solidHeader = TRUE,
      collapsible = TRUE,width = 9,
      verbatimTextOutput("TestTableRender_jrip")
    ),
    box(title = "Model Summary", solidHeader = TRUE,
      collapsible = TRUE,width = 9,
      verbatimTextOutput("TestSummaryRender_jrip")
    )
  )
),
##### KMeans
#####
tabItem(tabName = "kmeans",
  fluidRow(
    box(title = "Data", solidHeader = TRUE,
      collapsible = TRUE,width = 9,
      verbatimTextOutput("submit_input_sample_kmeans")
    ),
    box(title = "Data Summary", solidHeader = TRUE,
      collapsible = TRUE,width = 9,
      verbatimTextOutput("TestSummaryRender_kmeans")
    ),
    box(width=2,
      uiOutput("dependents_delcol_kmeans",
        style = "overflow-y:scroll; max-height: 100px; background:
ghostwhite;"),
      uiOutput("dependents_selcol_kmeans"),
      numericInput("seed", "Seed", 1),
      numericInput("k_mv", "Centers", 2,min=2),
      numericInput("k_nstart", "nstart", 1),
      uiOutput("dependents_button_kmeans")
    ),
    box(title = "Cross Table", solidHeader = TRUE,
      collapsible = TRUE,width = 9,
      verbatimTextOutput("TestTableRender_kmeans")
    ),
  )
),

```

```

        box(title = "Model Centers", solidHeader = TRUE,
            collapsible = TRUE,width = 9,
            verbatimTextOutput("TestCenterRender_kmeans")
        ),
        box(title = "Model Plot", solidHeader = TRUE,
            collapsible = TRUE,collapsed=TRUE,width = 9,
            plotOutput("plot_kmeans",height='auto')
        )
    )
),
##### knn #####
tabItem(tabName = "knn",
    fluidRow(
        box(
            title = "Data", solidHeader = TRUE,
            collapsible = TRUE,width = 9,
            verbatimTextOutput("submit_input_sample_knn")
        ),
        box(
            title = "Data Summary", solidHeader = TRUE,
            collapsible = TRUE,width = 9,
            verbatimTextOutput("TestSummaryRender_knn")
        ),
        box(
            width=2,
            uiOutput("dependents_delcol_knn",
                style = "overflow-y:scroll; max-height: 100px; background: ghostwhite;"),
            uiOutput("dependents_selcol_knn"),
            numericInput("knn_k", "k", 'Default',min=2),
            uiOutput("dependents_button_knn")

        ),
        box(
            title = "Cross Table", solidHeader = TRUE,
            collapsible = TRUE,width = 9,
            verbatimTextOutput("TestTableRender_knn")
        )
    )
),

##### naive bayes #####
tabItem(tabName = "naive",
    fluidRow(
        box(
            title = "Data", solidHeader = TRUE,
            collapsible = TRUE,width = 9,
            verbatimTextOutput("submit_input_sample_naive")
        ),
        box(

```

```

width=2,
uiOutput("dependents_delcol_naive",
        style = "overflow-y:scroll; max-height: 100px; background: ghostwhite;"),
uiOutput("dependents_selcol_naive"),
numericInput("laplace", "Laplace", 0, min = 0, max = 1),
numericInput("seed", "Seed", 1),
uiOutput("dependents_button_naive")
),
box(
  title = "Cross Table", solidHeader = TRUE,
  collapsible = TRUE,width = 9,
  verbatimTextOutput("TestTableRender_naive")
),
box(
  title = "Model", solidHeader = TRUE,
  collapsible = TRUE,width = 9,
  verbatimTextOutput("TestModelRender_naive")
)
)
),

```

```

##### neuralnet
#####
tabItem(tabName = "neuralnet",
  fluidRow(
    box(
      title = "Data", solidHeader = TRUE,
      collapsible = TRUE,width = 9,
      verbatimTextOutput("submit_input_sample_neuralnet")
    ),
    box(
      width=2,
      uiOutput("dependents_delcol_neuralnet",
              style = "overflow-y:scroll; max-height: 100px; background: ghostwhite;"),
      uiOutput("dependents_selcol_neuralnet"),
      uiOutput("dependents_type_neuralnet"),
      numericInput("seed", "Seed:", 1),
      numericInput("hidden_1", "Hidden Layer 1:", 3,min=1),
      numericInput("hidden_2", "Hidden Layer 2:", 2,min=1),
      uiOutput("dependents_button_neuralnet")
    ),
    box(
      title = "Accuracy", solidHeader = TRUE,
      collapsible = TRUE, width = 2,
      verbatimTextOutput("TestAccuracyRender_neuralnet")
    ),
    box(
      title = "Predict", solidHeader = TRUE,

```

```

        collapsible = TRUE, width = 7,
        verbatimTextOutput("TestTableRender_neuralnet")
    ),
    box(
        title = "Model Plot", solidHeader = TRUE,
        collapsible = TRUE,width = 9,
        plotOutput("plot_neural",height='auto')
    )
)

),
##### bar plot
#####
    tabItem(tabName = "barplot",
        sidebarPanel(
            selectInput("in_sel_bar_yVar","y Variable:", choices = NULL),
            selectInput("in_sel_bar_xVar","x Variable:", choices = NULL)
        ),
        mainPanel(
            plotOutput('plot_bar')
        )
    ),
##### pie chart
#####
    tabItem(tabName = "piechart",
        sidebarPanel(
            selectInput("in_sel_pie_xVar","x Variable:", choices = NULL),
            selectInput("in_sel_pie_yVar","y Variable:", choices = NULL)

        ),
        mainPanel(
            plotlyOutput('plot_pie')
        )
    ),
##### line graph
#####
    tabItem(tabName = "lineplot",
        sidebarPanel(
            selectInput("in_sel_line_yVar","y Variable:", choices = NULL),
            selectInput("in_sel_line_xVar","x Variable:", choices = NULL)

        ),
        mainPanel(
            plotlyOutput('plot_line')
        )
    ),
##### scatter plot
#####
    tabItem(tabName = "scatterplot",
        sidebarPanel(

```

```

        selectInput("in_sel_scatter_yVar","y Variable:", choices = NULL),
        selectInput("in_sel_scatter_xVar","x Variable:", choices = NULL)

    ),
    mainPanel(
        plotOutput('plot_scatter'),
        textOutput('text_scatter')
    )
),
##### box plot
#####
    tabItem(tabName = "boxplot",
        sidebarPanel(
            selectInput("in_sel_box_xVar","x Variable:", choices = NULL)

        ),
        mainPanel(
            plotOutput('plot_box'),
            verbatimTextOutput('text_box')
        )
    ),
##### pairs #####
    tabItem(tabName = "pairs",
        sidebarPanel(
            uiOutput("dependents_delcol_pairs",
                style = "overflow-y:scroll; max-height: 100px; background: ghostwhite;"),
            uiOutput("dependents_selcol_pairs",title='Select Colour'),
            numericInput("alpha", "Alpha", 0.5),
            uiOutput("dependents_button_pairs"),
            width=2
        ),
        mainPanel(
            plotOutput('plot_pairs')
        )
    )
)

)
)

##### ui #####
ui<-dashboardPage(
    skin='black',
    header,
    sidebar,
    body
)

##### 3. Server #####

server <- function(input, output,session) {

```

```
options(warn = -1)
options(shiny.maxRequestSize = 30*1024^2)
```

```
dataload<-reactive({
  req(input$file1)

  file1 = input$file1
  data1 = read.csv(file1$datapath)

  updateSelectInput(session, "in_sel_bar_xVar", choices = colnames(data1))
  updateSelectInput(session, "in_sel_bar_yVar", choices = colnames(data1))

  updateSelectInput(session, "in_sel_pie_xVar", choices = colnames(data1))
  updateSelectInput(session, "in_sel_pie_yVar", choices = colnames(data1))

  updateSelectInput(session, "in_sel_line_xVar", choices = colnames(data1))
  updateSelectInput(session, "in_sel_line_yVar", choices = colnames(data1))

  updateSelectInput(session, "in_sel_scatter_xVar", choices = colnames(data1))
  updateSelectInput(session, "in_sel_scatter_yVar", choices = colnames(data1))

  num <- c()
  for (i in seq(ncol(data1))) {
    if (is.numeric(data1[1,i])==T) num <- c(num,colnames(data1)[i])
  }
  updateSelectInput(session, "in_sel_box_xVar", choices = num)

  return(data1)
})
```

```
len <- function(){
  dl <- dataload()
  return(ceiling(ncol(dl)/3)*400)
}
```

```
####table_format
output$table <- DT::renderDataTable(DT::datatable({
  req(input$file1)

  file1 = input$file1
  data1 = read.csv(file1$datapath)

}))
```

```
#### statistic
#### summary
quant <- function(dt){
```

```

q <- c()
name <- c()
for (i in seq(ncol(dt))){
  if(is.numeric(dt[1,i]) == TRUE){
    name <- c(name,colnames(dt[i]))
    quan <- quantile(unlist(dt[i]),probs = seq(0.1,1,0.1),na.rm=T)

    sm <- c()
    for (j in seq(length(quan))){
      if(j==1){
        sm <- rbind(sm, sum(unlist(dt[i]) <= quan[j]))
      }
      else {
        sm <- rbind(sm,sum(unlist(dt[i]) > quan[j-1] & unlist(dt[i]) <= quan[j] ))
      }
    }
    q <- cbind(q,sm)
  }
}
df <- data.frame(q)
colnames(df) <- name
rownames(df) <- c('Q01','Q02','Q03','Q04','Q05','Q06','Q07','Q08','Q09','Q10')
return(df)
}

output$stat_summary <- renderPrint({
  dl <- dataload()
  stat.summary <- function(dt) {
    v <- c()
    for (i in seq(ncol(dt))){
      if (is.numeric(dt[1,i])==TRUE){
        a <- paste('Min   : ', sprintf('%.4f', min(dt[i])))
        b <- paste('Mean   : ', sprintf('%.4f', mean(unlist(dt[i])),4))
        c <- paste('Median : ', sprintf('%.4f', median(unlist(dt[i]))))
        d <- paste('Max    : ', sprintf('%.4f', max(dt[i])))
        e <- paste('SD     : ', sprintf('%.4f', sd(unlist(dt[i]))))
        g <- paste('NA     : ', sum(is.na(dt[i])))

        f <- rbind(a,b,c,d,e,g)
      }
      else{
        na <- paste('NA     : ', round(sum(is.na(dt[i]))))
        f <- c(summary(dt[i]),na)
      }
    }

    w <- max(nchar(f))

    if(i == 1){
      v <- format(f,width=w)
    }
    else{
      if (length(v) < length(f)){
        v <- rbind(v,matrix('',nrow=nrow(f)-nrow(v),ncol=ncol(v)))
      }
    }
  }
})

```

```

    }
    else if (nrow(v) > length(f)){
      f <- c(f,rep("",nrow(v)-length(f)))
    }

    v <- cbind(v,format(f,width=w))
  }

}
r <- data.frame(v)
colnames(r) <- c(colnames(dt))
rownames(r) <- c()
return(data.frame(r))
}
data.table(stat.summary(dl))
})
output$quan <- renderPrint({
  dl <- dataload()
  data.table(quant(dl))
})
output$plot_quan <- renderPlot({
  dl <- dataload()
  bar <- function(rdt){
    par(mfrow=c(ceiling(ncol(rdt)/3),3),mar=c(3,3,3,3),pty = "s")
    colfunc <- colorRampPalette(c("cadetblue1", "darkseagreen1"))
    dt <- quant(rdt)
    k <- 1
    for (i in seq(ncol(rdt))){
      dumb <- c()
      if (is.numeric(unlist(rdt[i])) == T){

        for(j in seq(nrow(dt[k]))){
          dumb <- c(dumb,rep(rownames(dt[k])[j],(unlist(dt[k])[j]+1)))
        }
        tb <- table(dumb)

        tt <- paste(colnames(dt)[k], 'Quantile Dist')
        barplot(tb,ylim=c(0,max(tb)+1),main = tt,col=colfunc(10),border=F,cex.main=
2,cex.names = 1.2,axes=F)

        k <- k+1
      }
    }
  }
  else{
    dumb <- c()
    unqn <- length(unique(unlist(rdt[i])))
    chr <- unique(rdt[i])[1]
    for(j in seq(unqn)){
      unq <- unique(rdt[i])[j,1]
      dumb <- c(dumb,rep(as.character(chr[j,1]),sum(unlist(rdt[i])==unq)))
    }
    tb <- table(dumb)
    tt <- paste(colnames(rdt)[i], 'Dist')
  }
}

```



```

        barplot(tb,ylim=c(0,max(tb)+1),main = tt,col=colfunc(unqn),border=F,cex.main=
2,cex.names = 2,axes=F)
    }
}

```

```

}
bar(dl)

},height=len)
output$plot_dist <- renderPlot({
  dl <- dataload()
  distplot <- function(dt){

    grid.newpage()
    pushViewport(viewport(layout = grid.layout(ceiling(ncol(dt)/3),3)))

    vplayout <- function(x, y) viewport(layout.pos.row = x, layout.pos.col = y)
    k <- 1
    l <- 1
    for (i in seq(ncol(dt))){
      if (is.numeric(unlist(dt[i])) == T){
        plt <- ggplot(dt, aes_string(colnames(dt)[i])) +
          geom_histogram(aes(y=..density..),binwidth=.5,
            colour="cadetblue4", fill="white") +
          geom_density(alpha=.2, fill="indianred1") + theme_bw()
      }
      else{
        plt <- ggplot(dt, aes_string(colnames(dt)[i])) +
          geom_bar(aes(y=..count..),stat='count',colour="cadetblue4", fill="white") +
theme_bw()
      }
    }
  }
}

```

```

    print(plt,vp = vplayout(k,l))
    l <- l + 1
    if(i%%3 == 0){
      k <- k + 1
      l <- 1
    }
  }
}
distplot(dl)

```

```

},height=len)

```

```

#### regression

```

```

output$dependents_delcol_regression <- renderUI({
  data <- dataload()
  if (is.null(data)) return(NULL)
  checkboxGroupInput(inputId = 'in_che_delcol_regression',
    label = "Delete colmun",
    choices = colnames(data),

```

```

        selected = 'null',
        inline = FALSE
    )
})
output$dependents_selcol_regression <- renderUI({
  data <- dataload()
  if (is.null(data)) return(NULL)
  if (length(input$in_che_delcol_regression) == 0){
    ch <- colnames(data)
  }
  else{
    ch <- colnames(data)[-which(colnames(data) == input$in_che_delcol_regression)]
  }
  selectInput("in_sel_label_regression", "Target", choices = ch)
})
output$dependents_button_regression <- renderUI({
  data <- dataload()
  if (is.null(data)) return(NULL)
  actionButton("in_btn_submit_regression", "Submit")
})

subinput_table_regression <- eventReactive(input$in_btn_submit_regression, {
  req(input$file1)
  file1 = input$file1
  data = read.csv(file1$datapath, stringsAsFactors = T)
  data1 <- data[!(colnames(data) %in% input$in_che_delcol_regression )]
  reg_data <- na.omit(data1)
  input_label <- as.formula(paste(colnames(reg_data[input$in_sel_label_regression]), "~."))

  model <- lm(input_label, data=reg_data)
  return(model)
})
output$submit_input_sample_regression <- renderPrint({
  req(input$file1)
  file1 = input$file1
  data <- read.csv(file1$datapath)
  data1 <- data[!(colnames(data) %in% input$in_che_delcol_regression )]
  return(head(data1, 7))
})
output$anova_Render_regression <- renderPrint({
  anova(subinput_table_regression())
})
output$coefficient_Render_regression <- renderPrint({
  summary(subinput_table_regression())
})
output$interval_Render_regression <- renderPrint({
  confint(subinput_table_regression())
})
output$plot_reg <- renderPlot({
  par(mfrow=c(2,2), pty = "s")
  plot(subinput_table_regression())
}, height=800)

```

```

## decision
output$dependents_delcol_decision <- renderUI({
  data <- dataload()
  if (is.null(data)) return(NULL) # 불필요한 컬럼 삭제시 작동되는 코드
  checkboxGroupInput(inputId = 'in_che_delcol_decision',
    label = "Delete colmun",
    choices = colnames(data),
    selected = 'null',
    inline = FALSE
  )
})
output$dependents_selcol_decision <- renderUI({
  data <- dataload()
  if (is.null(data)) return(NULL)
  if (length(input$in_che_delcol_decision) == 0){
    ch <- colnames(data)
  }
  else{
    ch <- colnames(data)[-which(colnames(data) == input$in_che_delcol_decision)]
  }
  selectInput("in_sel_label_decision", "Target", choices = ch)
})
output$dependents_selmodel_decision <- renderUI({
  data <- dataload()
  if (is.null(data)) return(NULL)
  selectInput("in_sel_model_decision", "Model", choices = c('Tree', 'Rules'))
})
output$dependents_selwinnow_decision <- renderUI({
  data <- dataload()
  if (is.null(data)) return(NULL)
  selectInput("in_sel_winnow_decision", "Winnow", choices = c(FALSE, TRUE))
})
output$dependents_button_decision <- renderUI({
  data <- dataload()
  if (is.null(data)) return(NULL)
  actionButton("in_btn_submit_decision", "submit")
})

output$submit_input_sample_decision <- renderPrint({
  req(input$file1)
  file1 = input$file1
  data <- read.csv(file1$datapath)
  data1 <- data[,!(colnames(data) %in% input$in_che_delcol_decision )]
  return(head(data1,5))
})
subinput_table_decision <- eventReactive(input$in_btn_submit_decision, {
  req(input$file1)
  file1 = input$file1
  data = read.csv(file1$datapath, stringsAsFactors = T)
  data1 <- data[,!(colnames(data) %in% input$in_che_delcol_decision )]
  data1 <- na.omit(data1)
})

```

```

train_cnt <- round(0.9 * nrow(data1))
set.seed(input$seed)
train_index <- sample(1:nrow(data1), train_cnt, replace = F)
train <- data1[train_index, ]
test <- data1[-train_index, ]
test_label <- factor(data1[-train_index, input$in_sel_label_decision])

if (input$in_sel_model_decision == 'Tree'){
  model <- C5.0(train[, -which(colnames(data1) == input$in_sel_label_decision)],
    factor(train[, input$in_sel_label_decision]), trials=input$trials ,
    control = C5.0Control(winnow = input$dependents_selwinnow_decision, seed
= input$seed))
}
else{
  model <- C5.0(train[, -which(colnames(data1) == input$in_sel_label_decision)],
    factor(train[, input$in_sel_label_decision]), trials=input$trials ,rules = TRUE,
    control = C5.0Control(winnow = input$dependents_selwinnow_decision, seed
= input$seed))
}

out <- list(model, test, test_label, colnames(data1))

return(out)
})
output$TestTableRender_decision <- renderPrint({
  result <- subinput_table_decision()
  model <- result[[1]]
  test <- result[[2]]
  test_label <- result[[3]]
  coln <- result[[4]]
  predict_result <- predict(model, test[, -which(coln == input$in_sel_label_decision)])
  CrossTable(test_label , predict_result)
})
output$TestSummaryRender_decision <- renderPrint({
  model <- subinput_table_decision()
  summary(model[[1]])
})

#### machine learning
#### jrip
output$dependents_delcol_jrip <- renderUI({
  data <- dataload()
  if (is.null(data)) return(NULL)
  checkboxGroupInput(inputId = 'in_che_delcol_jrip',
    label = "Delete Colmun",
    choices = colnames(data),
    selected = 'null',
    inline = FALSE

```

```

    )
  })
  output$dependents_selcol_jrip <- renderUI({
    data <- dataload()
    if (is.null(data)) return(NULL)
    if (length(input$in_che_delcol_jrip) == 0){
      ch <- colnames(data)
    }
    else{
      ch <- colnames(data)[-which(colnames(data) == input$in_che_delcol_jrip)]
    }
    selectInput("in_sel_label_jrip", "Target", choices = ch)
  })
  output$dependents_button_jrip <- renderUI({
    data <- dataload()      # submit 버튼 눌렀을때 작동되는 코드
    if (is.null(data)) return(NULL)
    actionButton("in_btn_submit_jrip", "Submit")
  })

  output$submit_input_sample_jrip <- renderPrint({
    req(input$file1)
    file1 = input$file1
    data <- read.csv(file1$datapath)
    data1 <- data[!(colnames(data) %in% input$in_che_delcol_jrip )]
    return(head(data1,5))
  })

  subinput_table_jrip <- eventReactive(input$in_btn_submit_jrip, {
    req(input$file1)
    file1 = input$file1
    data = read.csv(file1$datapath, stringsAsFactors = T)
    data1 <- data[!(colnames(data) %in% input$in_che_delcol_jrip )]
    data1 <- na.omit(data1)

    input_label <- as.formula(paste(colnames(data1[input$in_sel_label_jrip]), "~."))

    train_cnt <- round(0.75*dim(data1)[1])
    set.seed(input$seed)
    train_index <- sample(1:dim(data1)[1], train_cnt, replace = F)
    train <- data1[train_index, ]
    test <- data1[-train_index, ]
    test_label <- factor(data1[-train_index, input$in_sel_label_jrip])

    model <- JRip(input_label, data=train, control = Weka_control(O = input$numopt))

    out <- list(model, test, test_label, colnames(data1))

    return(out)
  })
  output$TestTableRender_jrip <- renderPrint({
    result <- subinput_table_jrip()
    model <- result[[1]]

```

```

test <- result[[2]]
test_label <- result[[3]]
coln <- result[[4]]
predict_result <- predict(model, test[, -which(coln == input$in_sel_label_jrip)])
CrossTable(test_label, predict_result)
})
output$TestSummaryRender_jrip <- renderPrint({
  model <- subinput_table_jrip()
  summary(model[[1]])
})

```

#### kmeans

```

output$dependents_delcol_kmeans <- renderUI({
  data <- dataload()
  if (is.null(data)) return(NULL)
  checkboxGroupInput(inputId = 'in_che_delcol_kmeans',
    label = "Delete Colmun",
    choices = colnames(data),
    selected = 'null',
    inline = FALSE
  )
})
output$dependents_selcol_kmeans <- renderUI({
  data <- dataload()
  if (is.null(data)) return(NULL)
  if (length(input$in_che_delcol_kmeans) == 0){
    ch <- colnames(data)
  }
  else{
    ch <- colnames(data)[-which(colnames(data) == input$in_che_delcol_kmeans)]
  }
  selectInput("in_sel_label_kmeans", "Target", choices = ch)
})
output$dependents_button_kmeans <- renderUI({
  data <- dataload()
  if (is.null(data)) return(NULL)
  actionButton("in_btn_submit_kmeans", "Submit")
})

```

```

output$submit_input_sample_kmeans <- renderPrint({
  req(input$file1)
  file1 = input$file1
  data <- read.csv(file1$datapath)
  data1 <- data[,!(colnames(data) %in% input$in_che_delcol_kmeans)]
  return(head(data1, 5))
})
output$TestSummaryRender_kmeans <- renderPrint({
  req(input$file1)
  file1 = input$file1
  data = read.csv(file1$datapath, stringsAsFactors = T)
  data1 <- data[,!(colnames(data) %in% input$in_che_delcol_kmeans)]

```

```

data1 <- na.omit(data1)
summary(data1)
})
subinput_table_kmeans <- eventReactive(input$in_btn_submit_kmeans, {

  set.seed(input$seed)

  req(input$file1)
  file1 = input$file1
  data = read.csv(file1$datapath,stringsAsFactors = T)
  data1 <- data[,!(colnames(data) %in% input$in_che_delcol_kmeans )]
  data1 <- na.omit(data1)

  input_label <- colnames(data1[input$in_sel_label_kmeans])

  data1_n <- data1[ , -which(colnames(data1)==input$in_sel_label_kmeans)]

  like_model <- kmeans( data1_n, input$k_mv ,nstart=input$k_nstart)

  out <- list(data1,like_model)

  return(out)
})
output$TestTableRender_kmeans <- renderPrint({
  result <- subinput_table_kmeans()
  data1 <- result[[1]]
  model <- result[[2]]
  x <- cbind(data1[ , which(colnames(data1)==input$in_sel_label_kmeans)] , model$cluster)
  x2 <- data.frame(x)
  table(x2,dnn=c('Test Data','Predict Result'))
})
output$TestCenterRender_kmeans <- renderPrint({
  result <- subinput_table_kmeans()
  model <- result[[2]]
  model$centers
})
output$plot_kmeans <- renderPlot({
  result <- subinput_table_kmeans()
  data <- result[[1]]
  model <- result[[2]]
  data_wo_label <- data[, -which(colnames(data)==input$in_sel_label_kmeans)]
  plot(data_wo_label,pch=model$cluster,col=model$cluster)
},height=len)

#### knn
output$dependents_delcol_knn <- renderUI({
  data <- dataload()
  if (is.null(data)) return(NULL)
  checkboxGroupInput(inputId = 'in_che_delcol_knn',
    label = "Delete Colmun",
    choices = colnames(data),
    selected = 'null',

```

```

        inline = FALSE
    )
})
output$dependents_selcol_knn <- renderUI({
  data <- dataload()
  if (is.null(data)) return(NULL)
  if (length(input$in_che_delcol_knn) == 0){
    ch <- colnames(data)
  }
  else{
    ch <- colnames(data)[-which(colnames(data) == input$in_che_delcol_knn)]
  }
  selectInput("in_sel_label_knn", "Target", choices = ch)
})
output$dependents_button_knn <- renderUI({
  data <- dataload()
  if (is.null(data)) return(NULL)
  actionButton("in_btn_submit_knn", "Submit")
})

output$submit_input_sample_knn <- renderPrint({
  req(input$file1)
  file1 = input$file1
  data <- read.csv(file1$datapath)
  data1 <- data[,!(colnames(data) %in% input$in_che_delcol_knn )]
  return(head(data1,5))
})

output$TestSummaryRender_knn <- renderPrint({
  req(input$file1)
  file1 = input$file1
  data = read.csv(file1$datapath, stringsAsFactors = T)
  data1 <- data[,!(colnames(data) %in% input$in_che_delcol_knn )]
  data1 <- na.omit(data1)
  summary(data1)
})

normalize <- function(x) {
  return (( x - min(x)) / (max(x) - min(x)))
}

subinput_table_knn <- eventReactive(input$in_btn_submit_knn, {
  req(input$file1)
  file1 = input$file1
  data = read.csv(file1$datapath, stringsAsFactors = FALSE)
  data1 <- data[,!(colnames(data) %in% input$in_che_delcol_kmeans )]
  data1 <- na.omit(data1)

  data1 <- as.data.frame(lapply(data1[, -which(colnames(data1) == input$in_sel_label_knn)],
normalize))

  train_index = as.integer(trunc(nrow(data1) * 0.8))

  train <- data1[1:as.integer(train_index), ]

```



```

test <- data1[as.integer(train_index+1):as.integer(nrow(data1)), ]

train_label <- data[1:as.integer(train_index),which(colnames(data)==input$in_sel_label_knn)]
test_label <- data[as.integer(train_index+
1):as.integer(nrow(data1)),which(colnames(data)==input$in_sel_label_knn) ]

train_label <- factor(train_label)

if (is.numeric(input$knn_k) == F){
  kk <- round(sqrt(length(train)))
}
else{
  kk <- input$knn_k
}

result <- knn(train=train , test=test , cl=train_label, k= kk)

cross_table <- CrossTable(test_label , result, prop.chisq=FALSE )

return(cross_table)

})
output$TestTableRender_knn <- renderPrint({
  subinput_table_knn()
})

#### naive bayes
output$dependents_delcol_naive <- renderUI({
  data <- dataload()
  if (is.null(data)) return(NULL)
  checkboxGroupInput(inputId = 'in_che_delcol_naive',
    label = "Delete Colmun",
    choices = colnames(data),
    selected = 'null',
    inline = FALSE
  )
})
output$dependents_selcol_naive <- renderUI({
  data <- dataload()
  if (is.null(data)) return(NULL)
  if (length(input$in_che_delcol_naive) == 0){
    ch <- colnames(data)
  }
  else{
    ch <- colnames(data)[-which(colnames(data) ==input$in_che_delcol_naive)]
  }
  selectInput("in_sel_label_naive","Target",choices = ch)
})
output$dependents_button_naive <- renderUI({
  data <- dataload()
  if (is.null(data)) return(NULL)
  actionButton("in_btn_submit_naive","Submit")
})

```

```

})

output$submit_input_sample_naive <- renderPrint({
  req(input$file1)
  file1 = input$file1
  data <- read.csv(file1$datapath)
  data1 <- data[,!(colnames(data) %in% input$in_che_delcol_naive )]
  return(head(data1,5))
})

subinput_table_naive <- eventReactive(input$in_btn_submit_naive, {
  req(input$file1)
  file1 = input$file1
  data = read.csv(file1$datapath,stringsAsFactors = T)
  data1 <- data[,!(colnames(data) %in% input$in_che_delcol_naive )]
  data1 <- na.omit(data1)

  input_label <- as.formula(paste(colnames(data1)[input$in_sel_label_naive], "~."))

  train_cnt <- round(0.7 * dim(data1)[(colnames(data) %in% input$in_sel_label_naive)])
  set.seed(input$seed)

  train_index <- sample(1:dim(data1)[(colnames(data) %in% input$in_sel_label_naive)],
train_cnt, replace = F)
  train <- data1[train_index, ]
  test <- data1[-train_index, ]

  model <- naiveBayes(input_label , data = train ,laplace = input$laplace )

  out <- list(model,colnames(data1),test)

  return(out)
})

output$TestTableRender_naive <- renderPrint({
  result <- subinput_table_naive()
  model <- result[[1]]
  coln <- result[[2]]
  test <- result[[3]]
  test_label <- test[, (coln %in% input$in_sel_label_naive )]
  predict_result <- predict(model, test[,!(coln %in% input$in_sel_label_naive )] )
  CrossTable(test_label , predict_result)
})

output$TestModelRender_naive <- renderPrint({
  result <- subinput_table_naive()
  result[[1]]
})

#### neuralnet
output$dependents_delcol_neuralnet <- renderUI({
  data <- data.load()
  if (is.null(data)) return(NULL)
  checkboxGroupInput(inputId = 'in_che_delcol_neuralnet',

```

```

        label    = "Delete Colmun",
        choices  = colnames(data),
        selected = 'null',
        inline   = FALSE
    )
})
output$dependents_selcol_neuralnet <- renderUI({
  data <- dataload()
  if (is.null(data)) return(NULL)
  if (length(input$in_che_delcol_naive) == 0){
    ch <- colnames(data)
  }
  else{
    ch <- colnames(data)[-which(colnames(data) == input$in_che_delcol_naive)]
  }
  selectInput("in_sel_label_neuralnet", "Target", choices = ch)
})
output$dependents_type_neuralnet <- renderUI({
  selectInput("in_sel_type_neuralnet", "Target Type", choices = c('Numeric', 'Character'))
})
output$dependents_button_neuralnet <- renderUI({
  data <- dataload()
  if (is.null(data)) return(NULL)

  actionButton("in_btn_submit_neuralnet", "Submit")
})

output$submit_input_sample_neuralnet <- renderPrint({
  req(input$file1)
  file1 = input$file1
  data <- read.csv(file1$datapath)
  data1 <- data[,!(colnames(data) %in% input$in_che_delcol_neuralnet )]
  return(head(data1,5))
})
subinput_table_neuralnet <- eventReactive(input$in_btn_submit_neuralnet, {
  req(input$file1)
  file1 = input$file1
  data = read.csv(file1$datapath, stringsAsFactors = T)
  data1 <- data[,!(colnames(data) %in% input$in_che_delcol_neuralnet )]
  data1 <- na.omit(data1)

  input_label <- as.formula(paste(colnames(data1)[input$in_sel_label_neuralnet], "~."))

  train_cnt <- round(0.75*dim(data1)[1])
  set.seed(input$seed)
  train_index <- sample(1:dim(data1)[1], train_cnt, replace = F)

  normalize <- function(x) {
    return ( (x-min(x)) / (max(x) - min(x)) )
  }

  if (input$in_sel_type_neuralnet == 'Numeric'){

```

```

    data1_norm <- as.data.frame(lapply(data1,normalize) )
  }
  else{
    data1_norm <- data.frame(data1[, input
$in_sel_label_neuralnet],scale(data1[,!(colnames(data) %in%input$in_sel_label_neuralnet)))]))
    colnames(data1_norm)[1] <- c(input$in_sel_label_neuralnet)
  }

  train <- data1_norm[train_index, ]
  test <- data1_norm[-train_index, ]

  test_label <- factor(data1_norm[-train_index, input$in_sel_label_neuralnet])

  model <- neuralnet(formula = input_label, data =train, hidden = c(input$hidden_1,input
$hidden_2) )

  out <- list(model,test,data1)

  return(out)
})
output$TestAccuracyRender_neuralnet <- renderPrint({
  result <- subinput_table_neuralnet()
  model <- result[[1]]
  test <- result[[2]]
  data <- result[[3]]
  if (input$in_sel_type_neuralnet == 'Numeric'){
    model_results <- neuralnet::compute(model, test[,!(colnames(test) %in%input
$in_sel_label_neuralnet)])
    results <- data.frame(actual = test[,input$in_sel_label_neuralnet], prediction =
model_results$net.result)
    predicted=results$prediction * abs(diff(range(data[,input$in_sel_label_neuralnet]))) +
min(data[,input$in_sel_label_neuralnet])
    actual=results$actual * abs(diff(range(data[,input$in_sel_label_neuralnet]))) +
min(data[,input$in_sel_label_neuralnet])
    comparison=data.frame(predicted,actual)
    deviation=((actual-predicted)/actual)
    comparison=data.frame(predicted,actual,deviation)
    accuracy=1-abs(mean(deviation))
    data.frame(Accuracy = accuracy,row.names="")
  }
  else{
    model_results <- neuralnet::compute(model, test[,!(colnames(test) %in%input
$in_sel_label_neuralnet)])
    predicted <- model_results$net.result
    predicted_label <- 0
    correct <- 0
    for(i in seq(nrow(predicted))){
      predicted_label <- which(predicted[i,]==max(predicted[i,]))
      if (as.numeric(as.factor(test[,input$in_sel_label_neuralnet]))[i]==predicted_label) correct
<- correct + 1
    }
    accuracy <- correct/nrow(test)
  }
})

```

```

    data.frame(Accuracy = accuracy,row.names=")
  }
})
output$TestTableRender_neuralnet <- renderPrint({
  result <- subinput_table_neuralnet()
  model <- result[[1]]
  test <- result[[2]]
  data <- result[[3]]
  if (input$in_sel_type_neuralnet == 'Numeric'){
    model_results <- neuralnet::compute(model, test[!(colnames(test) %in%input
$input_sel_label_neuralnet)])
    results <- data.frame(actual = test[,input$in_sel_label_neuralnet], prediction =
model_results$net.result)
    results
  }
  else{
    model_results <- neuralnet::compute(model, test[!(colnames(test) %in%input
$input_sel_label_neuralnet)])
    predicted <- model_results$net.result

    predicted_label <- c()
    for(i in seq(nrow(predicted))){
      predicted_label <- c(predicted_label,which(predicted[i,]==max(predicted[i,])))
    }

    CrossTable(unlist(test[,input
$input_sel_label_neuralnet]),predicted_label,dnn=c('test_label','predicted_label'))
  }

})
output$plot_neural <- renderPlot({
  result <- subinput_table_neuralnet()
  model <- result[[1]]
  dev.off()
  plot(model)
},height=600)

#### graph
#### bar_plot
output$plot_bar <- renderPlot({
  table_in<-dataload()

  xdata<-as.factor(table_in[,input$in_sel_bar_xVar])
  ydata<-as.factor(table_in[,input$in_sel_bar_yVar])
  fdata=data.frame(x=xdata,y=ydata)

  ggplot(fdata) +
    geom_bar(aes_string(x='x',y='y',fill='x'),stat = "identity",show.legend=F)

```

```

})
#### pie_chart
output$plot_pie <- renderPlotly({
  table_in<-dataload()

  plot_ly(table_in, labels = ~table_in[,input$in_sel_pie_xVar],
    values = ~table_in[,input$in_sel_pie_yVar],type='pie')

})
#### line_plot
output$plot_line <- renderPlotly({
  table_in<-dataload()

  x <- list(title = input$in_sel_line_xVar)
  y <- list(title = input$in_sel_line_yVar)

  plot_ly(data = table_in,x=~table_in[,input$in_sel_line_xVar],y=~table_in[,input
$in_sel_line_yVar],type='scatter',mode='dot')%>%
  layout(xaxis = x, yaxis = y)

})
#### scatter_plot
output$plot_scatter <- renderPlot({
  table_in<-dataload()

  xyplot(table_in[,input$in_sel_scatter_yVar]~table_in[,input$in_sel_scatter_xVar],
grid=T,type=c('p'),col.line='darkorange',lwd=2, xlab=input$in_sel_scatter_xVar,ylab=input
$in_sel_scatter_yVar)

})
#### scatter_cov
output$text_scatter <- renderText({
  table_in<-dataload()
  paste("The correlation between the two is: ", cor(table_in[,input
$in_sel_scatter_yVar],table_in[,input$in_sel_scatter_xVar]))
})
#### box_plot
output$plot_box <- renderPlot({
  table_in<-dataload()
  bwplot(table_in[,input$in_sel_box_xVar],xlab=input$in_sel_box_xVar)

})
#### box_summary
output$text_box <- renderPrint({
  table_in<-dataload()
  dataset <- table_in[,input$in_sel_box_xVar]
  summary(dataset)
})
#### pairs

```

```

output$dependents_delcol_pairs <- renderUI({
  data <- dataload()
  if (is.null(data)) return(NULL)
  checkboxGroupInput(inputId = 'in_che_delcol_pairs',
    label = "Delete colmun",
    choices = colnames(data),
    selected = 'null',
    inline = FALSE
  )
})
output$dependents_selcol_pairs <- renderUI({
  data <- dataload()
  if (is.null(data)) return(NULL)
  ch <- c('None')
  for (i in seq(colnames(data))){
    if (is.numeric(data[1,i])==F){
      ch <- c(ch,colnames(data)[i])
    }
  }
  selectInput("in_sel_label_pairs","Submit",label = 'Select Colour',choices = ch)
})
output$dependents_button_pairs <- renderUI({
  data <- dataload()
  if (is.null(data)) return(NULL)
  actionButton("in_btn_submit_pairs","Submit")
})
subinput_pairs <- eventReactive(input$in_btn_submit_pairs, {
  req(input$file1)
  file1 <- input$file1
  data <- read.csv(file1$datapath)
  data <- data[,!(colnames(data) %in% input$in_che_delcol_pairs )]

  if (input$in_sel_label_pairs == 'None'){
    pa <- ggpairs(data,aes_string(alpha=input$alpha))
  }
  else {
    pa <- ggpairs(data,aes_string(colour = input$in_sel_label_pairs, alpha=input$alpha))
  }

  return(pa)
})
output$plot_pairs <- renderPlot({
  subinput_pairs()
},height=len)
}

```

##### 4. 샤이니 실행 #####

```
shinyApp(ui = ui, server = server)
```

## ■ R shiny의 큰 틀 ?

1. ui <- { 화면구현 }
  - 1.1 sidebar
  - 1.2 body
2. server <- { 데이터를 분석하고 결과를 출력 구현 }
  1. 테이블 결과 출력하는 코드
  2. summary boxplot 구현
  3. 통계 구현( 회귀분석, 의사결정트리 )
  4. 머신러닝 구현( jriper, knn, 신경망 등 )
  5. 그래프 그리는 코드( 원형, 막대, 산포도, 박스 그래프 등 )

**회사 지원할 때 포트폴리오로 제출, 경진대회에 포트폴리오 제출 하고자 할때?**

1. 제목변경
2. 경진대회 또는 내가 분석하고자하는 데이터( 전처리하고 잘 가공해 놓은 것 )를 같이 첨부
3. 샤이니 사용해서 분석결과 PPT 3장
4. 홈페이지로 구현 ( 이력서에 홈페이지 url을 기술하기 위해서 )

ui.R과 server.R이 필요하다.

## ■ object detection ( 사물 검출 )

배경보다는 내가 분류하고자하는 그 사물에만 집중해서 신경망을 학습시킬 수는 없을까? 라는 아이디어에서 출발

**사진에서 특정 사물만 분류해낼 수 있는 기술로 발전**

**예: 007 영상에서 넥타이, 가방, 트럭 등을 분류하는 기술**

실습 : 코랩에서 수행

[오브젝트\\_디텍션.ipynb](#)

**문제. 다른 사진을 코랩에 올려서 object detection 하시오!**



# 03/26

2021년 3월 26일 금요일 오전 9:44

## ■ 동영상을 object detection 하는 방법

[오브젝트\\_디텍션\\_동영상\\_수정2.ipynb](#)

1. 동영상 내에서 특정 사물을 분류하기 위한 가중치( 1000개의 클래스 분류 )가 필요
2. 코덱( 현재 os에 동영상이 플레이 될 수 있도록 인코딩, 디코딩하는 코드 )을 다운로드 받아야 함 ( 코덱 os에 맞는 코덱을 받아야 함 )
3. 동영상 파일이 필요
4. 1000개의 클래스를 분류하는 가중치를 신경망에 셋팅해서 동영상에서 사물을 detection 하는 코드가 필요 ( detect.py )

문제. 쉬는 시간 동안 objec detection 하는 새로운 영상을 준비한다.

이력서 낼 때 사용할 비공개 유튜브 url을 만들 수 있도록 영상을 준비한다.

## ■ 블랙핑크 영상에서 지수만 찾아서 detection 하는 준비물

<https://cafe.daum.net/oracleoracle/SgRM/188>

11기 게시판의 yolo lisa

1. 영상1개
2. 사진1개
3. 구현 코드

03/29

2021년 3월 29일 월요일 오전 9:48

## ■ 블랙핑크 영상에서 지수만 찾아서 detection 하는 준비물

<https://cafe.daum.net/oracleoracle/SqRM/328> 참고

11기 게시판의 yolo lisa

1. 영상1개
2. 사진1개
3. 구현 코드

\* 환경구성:

( 일본사람 블로그 참조 )

\* 코드구현: 인도사람 코드

<https://www.analyticsvidhya.com/blog/2018/12/introduction-face-detection-video-deep-learning-python/>

사진 한장을 신경망에 넣고 학습해서 그 사진에 해당하는 인물의 얼굴을 영상에서 찾는 코드

카카오에서 만든 팟플레이어를 설치한다.

영상을 플레이하고 tab을 누르면 영상에 대한 정보를 알 수 있는데 거기서 영상 사이즈를 확인한다.

영상 생성하고 10분을 기다려준다. ( 구글에서 내부적으로 영상을 만드는 시간을 줘야한다. )

그리고 나서 다운로드 받는다.

[오브젝트\\_디텍션\\_동영상\\_수정3\\_sj.ipynb](#)

[videosuzy.mp4](#)

